Distributed Cache Service

Visão geral de serviço

Edição 01

Data 07-11-2022

HUAWEI TECHNOLOGIES CO., LTD.

Marcas registadas e permissões


e outras marcas registadas da Huawei são marcas registadas da Huawei Technologies Co., Ltd. Todos as outras marcas registadas e os nomes registados mencionados neste documento são propriedade dos seus respectivos detentores.

Aviso

Os produtos, serviços e funcionalidades adquiridos são estipulados pelo contrato feito entre a Huawei e o cliente. Todos ou parte dos produtos, serviços e funcionalidades descritos neste documento pode não estar dentro do âmbito de aquisição ou do âmbito de uso. Salvo especificação em contrário no contrato, todas as declarações, informações e recomendações neste documento são fornecidas "TAL COMO ESTÁ" sem garantias, ou representações de qualquer tipo, seja expressa ou implícita.

As informações contidas neste documento estão sujeitas a alterações sem aviso prévio. Foram feitos todos os esforços na preparação deste documento para assegurar a exatidão do conteúdo, mas todas as declarações, informações e recomendações contidas neste documento não constituem uma garantia de qualquer tipo, expressa ou implícita.

Índice

1 O que é o DCS?

O HUAWEI CLOUD Distributed Cache Service (DCS) é um serviço de cache on-line, distribuído e in-memory compatível com Redis e Memcached. Ele é confiável, escalável, utilizável fora da caixa e fácil de gerenciar, atendendo aos seus requisitos de alto desempenho de leitura/gravação e acesso rápido aos dados.

- Pronto para uso

  O DCS fornece instâncias de divisão de nó único, principal/em espera, cluster de proxy, cluster do Redis e leitura/gravação com especificações que variam de 128 MB a 1024 GB. As instâncias de DCS podem ser criadas com apenas alguns cliques no console, sem a necessidade de preparar os servidores.

  As instâncias do DCS Redis 4.0, 6.0, e 5.0 são em contêiner e podem ser criadas em segundos.

- Segurança e confiabilidade

  O armazenamento e o acesso a dados de instâncias são protegidos com segurança por meio dos serviços de gerenciamento de segurança da HUAWEI CLOUD, incluindo Identity and Access Management (IAM), Virtual Private Cloud (VPC), Cloud Eye e Cloud Trace Service (CTS).

  As instâncias principal/em espera e de cluster podem ser implantadas em uma zona de disponibilidade (AZ) ou entre AZs.

- Auto Scaling

  As instâncias do DCS podem ser ampliadas ou reduzidas online, ajudando você a controlar os custos com base nos requisitos de serviço.

- Gerenciamento fácil

  Um console baseado na Web é fornecido para você executar várias operações, como reiniciar instâncias, modificar parâmetros de configuração e fazer backup e restaurar dados. As interfaces de programação de aplicativos (APIs) RESTful também são fornecidas para o gerenciamento automático de instâncias.

- Migração online

  Você pode criar uma tarefa de migração de dados no console para importar arquivos de backup ou migrar dados online.

Para obter detalhes sobre como selecionar um mecanismo de cache, consulte **Comparando Redis e Memcached**.

DCS para Redis

O DCS for Redis oferece suporte ao Redis 3.0, 4.0, 5.0 e 6.0.

- HUAWEI CLOUD DCS for Redis 3.0, 4.0 e 5.0

  NOTA

  O DCS for Redis 3.0 não é mais fornecido. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.

  O Redis é um sistema de armazenamento que suporta vários tipos de estruturas de dados, incluindo pares de chave-valor. Ele pode ser usado em cenários como cache de dados, publicação/assinatura de eventos e enfileiramento de alta velocidade, conforme descrito em **Cenários de aplicação**. O Redis é escrito em ANSI C, suportando leitura/gravação direta de **strings**, **hashes**, **listas**, **conjuntos**, **conjuntos ordenados** e **fluxos**. O Redis trabalha com um conjunto de dados na memória que pode ser persistido no disco.

  As instâncias do DCS Redis podem ser personalizadas com base em suas necessidades.

Tabela 1-1 Configurações de instância do DCS Redis

Tipos de instância	O DCS for Redis fornece os seguintes tipos de instâncias para se adequar a diferentes cenários de serviço: Único-nó: Adequado para armazenamento em cache de dados temporários em cenários de baixa confiabilidade. As instâncias de nó único suportam operações de leitura/gravação altamente simultâneas, mas não suportam a persistência de dados. Os dados serão excluídos depois que as instâncias forem reiniciadas. Principal/Em espera: Cada instância principal/em espera é executada em dois nós (um principal e um stand-by). O nó stand-by replica os dados de forma síncrona a partir do nó principal. Se o nó principal falhar, o nó stand-by torna-se automaticamente o nó principal. Você pode dividir as operações de leitura e gravação escrevendo no nó principal e lendo a partir do nó stand-by. Isso melhora o desempenho geral de leitura/gravação do cache. Cluster de proxy Além do cluster Redis nativo, uma instância de cluster de proxy tem proxies e balanceadores de carga. Os balanceadores de carga implementam o balanceamento de carga. Diferentes solicitações são distribuídas para diferentes proxies para alcançar alta simultaneidade. Cada estilhaço no cluster tem um nó principal e um nó stand-by. Se o nó principal estiver com defeito, o nó em em espera no mesmo estilhaço será promovido à função de principal para assumir os serviços. Cluster do Redis Cada instância do Cluster do Redis consiste em vários **estilhaços** e cada estilhaço inclui um nó principal e várias réplicas (ou nenhuma réplica). Fragmentos não são visíveis para você. Se o nó principal falhar, uma réplica no mesmo estilhaço assumirá os serviços. Você pode dividir as operações de leitura e gravação escrevendo no nó principal e lendo a partir das réplicas. Isso melhora o desempenho geral de leitura/gravação do cache. Separação de leitura/gravação Uma instância de divisão de leitura/gravação tem proxies e balanceadores de carga, além da arquitetura principal/em espera. Os balanceadores de carga implementam o balanceamento de carga e diferentes solicitações são distribuídas para diferentes proxies. Os proxies distinguem entre solicitações de leitura e gravação e as enviam para nós principais ou nós em espera, respectivamente.
Especificações da instância	O DCS for Redis fornece instâncias de especificações diferentes, variando de 128 MB a 1024 GB.
Compatibilidade de código aberto	As instâncias do DCS são compatíveis com o Redis 3.0, 4.0, 5.0 e 6.0 de código aberto.
Arquitetura subjacente	Redis **Standard** baseado em VMs: suporta até 100 000 consultas por segundo (QPS) em um único nó.

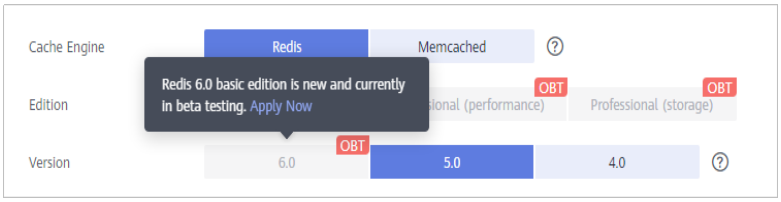
Alta disponibilidade (HA) e recuperação de desastres (DR)	Todas as instâncias, exceto as de nó único, podem ser implantadas em AZs em uma região com fontes de alimentação e redes fisicamente isoladas.

Para obter mais informações sobre o Redis de código aberto, visite **https://redis.io/**.

- HUAWEI CLOUD DCS para Redis 6.0 (OBT)

  NOTA

  O Redis 6.0 está em OBT. Você pode acessar a página para criar uma instância no console do DCS e clicar em **Apply Now** para solicitar a permissão OBT, conforme mostrado na figura a seguir.

  O HUAWEI CLOUD DCS for Redis agora apresenta alto desempenho baseado em multithreading.

  O alto desempenho baseado em multithreading é alcançado com base no KeyDB de código aberto, que é um fork do Redis que oferece alto desempenho. O KeyDB se concentra em multithreading, eficiência de memória e alto throughput, e fornece recursos encontrados apenas no Redis Enterprise. O KeyDB é totalmente compatível com o protocolo Redis, módulos e scripts, e garante a atômicaidade de scripts e transações. O desenvolvimento do KeyDB acompanha o do Redis, portanto, o KeyDB fornece um superconjunto de funcionalidades do Redis e pode substituir as implantações existentes do Redis. Com o mesmo hardware, o KeyDB processa duas vezes mais consultas por segundo do que o Redis, com uma latência 60% menor. A replicação ativa simplifica o failover de hot em espera, permitindo que você aloque facilmente operações de gravação em réplicas e use o balanceamento de carga ou failover simples baseado em TCP. O alto desempenho do KeyDB permite que você consiga mais com menos hardware, reduzindo custos de operação e complexidade.

  No KeyDB, I/O e loops de eventos são executados em vários threads. O KeyDB suporta recursos como expiração de subchaves, múltiplos principais, hashes aninhados e scripts CRON para Lua, que não estão disponíveis no Redis.

  Em versões anteriores ao Redis 6.0, uma consulta lenta geralmente faz com que outras consultas sejam atrasadas devido ao modelo de thread único. Para resolver problemas de desempenho, a nova edição fez uma série de otimização com base em um modelo multi-thread. A simultaneidade multi-thread foi aprimorada para processamento de eventos de I/O e back-end; o acesso aos dados em cache é ainda mais acelerado por meio de spinlock justo; as chaves expiradas podem ser removidas duas vezes mais rapidamente graças a algoritmos otimizados; O suporte para subchave expira também melhora o desempenho de leitura/gravação de chaves grandes. Como resultado, a nova edição é adequada em cenários que exigem alto desempenho de nó único, como tópicos de tendências e eventos de transmissão ao vivo na Internet.

As instâncias do DCS Redis podem ser personalizadas com base em suas necessidades.

Tabela 1-2 Configurações de instância do DCS Redis

Tipos de instância	O DCS for Redis fornece os seguintes tipos de instâncias para se adequar a diferentes cenários de serviço: Único-nó: Adequado para armazenamento em cache de dados temporários em cenários de baixa confiabilidade. As instâncias de nó único suportam operações de leitura/gravação altamente simultâneas, mas não suportam a persistência de dados. Os dados serão excluídos depois que as instâncias forem reiniciadas. Principal/Em espera: Cada instância principal/em espera é executada em dois nós (um principal e um em espera). O nó em espera replica os dados de forma síncrona a partir do nó principal. Se o nó principal falhar, o nó em espera torna-se automaticamente o nó principal. Você pode dividir as operações de leitura e gravação escrevendo no nó principal e lendo a partir do nó em espera. Isso melhora o desempenho geral de leitura/gravação do cache. Cluster de proxy Além do cluster Redis nativo, uma instância de cluster de proxy tem proxies e balanceadores de carga. Os balanceadores de carga implementam o balanceamento de carga. Diferentes solicitações são distribuídas para diferentes proxies para alcançar alta simultaneidade. Cada estilhaço no cluster tem um nó principal e um nó em espera. Se o nó principal estiver com defeito, o nó em espera no mesmo estilhaço será promovido à função de principal para assumir os serviços. Cluster do Redis Cada instância do Cluster do Redis consiste em vários **estilhaços** e cada estilhaço inclui um nó principal e várias réplicas (ou nenhuma réplica). Fragmentos não são visíveis para você. Se o nó principal falhar, uma réplica no mesmo estilhaço assumirá os serviços. Você pode dividir as operações de leitura e gravação escrevendo no nó principal e lendo a partir das réplicas. Isso melhora o desempenho geral de leitura/gravação do cache. Separação de leitura/gravação Uma instância de divisão de leitura/gravação tem proxies e balanceadores de carga, além da arquitetura principal/em espera. Os balanceadores de carga implementam o balanceamento de carga e diferentes solicitações são distribuídas para diferentes proxies. Os proxies distinguem entre solicitações de leitura e gravação e as enviam para nós principal ou nós em espera, respectivamente.
Especificações da instância	Faixa de 4 GB a 8 GB, 16 GB, 32 GB e 64 GB.
Compatibilidade de código aberto	KeyDB 6.0.16

Arquitetura subjacente	Redis **Standard** baseado em VMs: suporta até 300 000 consultas por segundo (QPS) em um único nó.
HA e DR	Todas as instâncias, exceto as de nó único, podem ser implantadas em AZs em uma região com fontes de alimentação e redes fisicamente isoladas.

Para obter mais informações sobre o KeyDB, visite **https://keydb.dev/**.

NOTA

A edição básica do DCS for Redis 6.0 fornece apenas instâncias principal/em espera e de nó único. As edições Enterprise (desempenho) e Enterprise (armazenamento) fornecem apenas instâncias principal/em espera.

DCS para Memcached (indisponível em breve)

NOTA

O DCS for Memcached está prestes a ficar indisponível e não é mais vendido em algumas regiões. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.

Memcached é um sistema de cache de valores-chave na memória que suporta leitura/gravação de strings simples. Ele é frequentemente usado para armazenar em cache dados de banco de dados de back-end para aliviar a carga nesses bancos de dados e acelerar aplicativos da Web. Para obter detalhes sobre seus cenários de aplicação, consulte **Cenários da Aplicação Memcached**.

Além da compatibilidade total com o Memcached, o DCS for Memcached fornece o hot em espera e a persistência de dados.

Tabela 1-3 Configuração da instância do Memcached DCS

Tipos de instância	O DCS para Memcached fornece os dois tipos de instâncias a seguir para atender a diferentes cenários de serviço: Único-nó: Adequado para armazenamento em cache de dados temporários em cenários de baixa confiabilidade. As instâncias de nó único suportam operações de leitura/gravação altamente simultâneas, mas não suportam a persistência de dados. Os dados serão excluídos depois que as instâncias forem reiniciadas. Principal/Em espera: Cada instância principal/em espera é executada em dois nós (um principal e um em espera). O nó em espera replica dados de forma síncrona a partir do nó principal, mas não suporta operações de leitura/gravação. Se o nó principal falhar, o nó em espera torna-se automaticamente o nó principal.
Memória	Especificação de instâncias de memcached de DCS de nó único ou principal/em espera: 2 GB, 4 GB, 8 GB, 16 GB, 32 GB e 64 GB.

HA e DR	As instâncias do Memcached DCS principal/em espera podem ser implantadas em AZs na mesma região com fontes de alimentação e redes fisicamente isoladas.

Para obter mais informações sobre o Memcached de código aberto, visite **https://memcached.org/**.

Introdução ao Vídeo DCS

Assista ao vídeo a seguir para saber mais sobre o DCS.

Distributed Cache Service

2 Cenários de aplicação

Cenários de aplicativos do Redis

Muitos sites de comércio eletrônico de grande escala e aplicativos de transmissão de vídeo e jogos exigem acesso rápido a grandes quantidades de dados que possuem estruturas de dados simples e não precisam de consultas frequentes. Nesses cenários, você pode usar o Redis para obter acesso rápido e barato aos dados. O Redis permite que você recupere dados de armazenamentos de dados na memória, em vez de confiar inteiramente em bancos de dados baseados em disco mais lentos. Além disso, você não precisa mais executar tarefas de gerenciamento adicionais. Esses recursos tornam o Redis um complemento importante aos bancos de dados tradicionais baseados em disco e um serviço básico essencial para aplicativos da Internet que recebem acesso de alta simultaneidade.

Os cenários típicos de aplicativos do DCS for Redis são os seguintes:

1. **E-commerce flash sales**

   Catálogo de produtos de comércio eletrônico, ofertas e dados de vendas em flash podem ser armazenados em cache no Redis.

   Por exemplo, o acesso a dados de alta concorrência em vendas flash dificilmente pode ser tratado por bancos de dados relacionais tradicionais. Isso requer que o hardware tenha uma configuração mais alta, como I/O de disco. Por outro lado, o Redis suporta QPS 100.000 por nó e permite implementar o bloqueio usando comandos simples como **SET**, **GET**, **DEL**, e **RPUSH** para lidar com vendas flash.

   Para obter detalhes sobre bloqueio, consulte **Implementing Bloqueios Distribuídos com Redis** nas *Melhores Práticas*.

2. **Live video commenting**

   Na transmissão ao vivo, os dados de usuários online, classificação de presentes e comentários de marcadores podem ser armazenados como conjuntos classificados no Redis.

   Por exemplo, comentários de marcadores podem ser devolvidos utilizando o comando **ZREVRANGEBYSCORE**. Os comandos **ZPOPMAX** e **ZPOPMIN** no Redis 5.0 podem facilitar ainda mais o processamento de mensagens.

3. **Game leaderboard**

   Nos jogos online, os jogadores mais bem classificados são exibidos e atualizados em tempo real. O ranking da tabela de classificação pode ser armazenado como conjuntos classificados, que são fáceis de usar com até 20 comandos.

   Para obter detalhes, consulte **Ranking com Redis** em *práticas recomendadas*.

4. **Social networking comments**

   Em aplicações web, consultas de comentários post geralmente envolvem classificação por tempo em ordem decrescente. À medida que os comentários se acumulam, a classificação se torna menos eficiente.

   Usando listas no Redis, um número predefinido de comentários pode ser retornado do cache, em vez de do disco, facilitando a carga do banco de dados e acelerando as respostas do aplicativo.

Cenários da Aplicação Memcached

O memcached é adequado para armazenar dados simples de chave-valor.

1. Páginas da Web

   O armazenamento em cache de dados estáticos, como páginas HTML, Cascading Style Sheets (CSS) e imagens em instâncias do DCS Memcached, melhora o desempenho de acesso de páginas da Web.

2. Banco de dados frontend

   Em sistemas dinâmicos, como redes sociais e sites de blogs, as operações de escrita são muito menores do que as operações de leitura, como consultar usuários, amigos e artigos. Para facilitar a carga do banco de dados e melhorar o desempenho, os seguintes dados podem ser armazenados em cache no Memcached:

   - Dados acessados com frequência que não exigem atualizações em tempo real e podem expirar automaticamente

     Exemplo: listas de artigos mais recentes e rankings. Embora os dados sejam gerados constantemente, seu impacto na experiência do usuário é limitado. Tais dados podem ser armazenados em cache por um período de tempo pré-definido e acessados a partir do banco de dados após esse período. Se os editores de páginas da Web desejarem visualizar a classificação mais recente, uma política de limpeza ou atualização de cache poderá ser configurada.

   - Dados acessados com frequência que exigem atualizações em tempo real

     Exemplo: listas de amigos, listas de artigos e registros de leitura. Esses dados podem ser armazenados em cache no Memcached primeiro e, em seguida, atualizados sempre que ocorrerem alterações (adicionando, modificando e excluindo dados).

3. Oferta relâmpago

   É difícil para os bancos de dados tradicionais gravar uma operação de colocação de pedidos durante as vendas flash no banco de dados, modificar os dados de inventário e garantir a consistência da transação, garantindo uma experiência ininterrupta do usuário.

   Os comandos memcached **incr** e **decr** podem ser usados para armazenar informações de inventário e completar a colocação de pedidos na memória. Uma vez que um pedido é enviado, um número de pedido é gerado. Em seguida, o pedido pode ser pago.

NOTA

Cenários em que o Memcached não é adequado:

- O tamanho de um único objeto de cache é maior que 1 MB.

  O memcached não pode armazenar em cache um objeto maior que 1 MB. Nesses casos, use o Redis.

- A chave contém mais de 250 caracteres.

  Para usar o Memcached em tal cenário, você pode gerar um hash MD5 para a chave e armazenar em cache o hash em vez disso.

- É necessária uma alta confiabilidade de dados.

  O Memcached de código aberto não fornece replicação, backup e migração de dados, portanto, a persistência de dados não é suportada.

  As instâncias do Memcached DCS principal/em espera oferecem suporte à persistência de dados. Para obter mais informações, entre em contato com o suporte técnico.

- Estruturas complexas de dados e processamento são necessários.

  O Memcached suporta apenas pares de chave-valor simples e não suporta estruturas de dados complexas, como listas e conjuntos, ou operações complexas, como classificação.

3 Tipos de instância do DCS

3.1 Redis de nó único

Três versões do Redis estão disponíveis para instâncias do Redis DCS de nó único: Redis 3.0, Redis 4.0 e Redis 5.0.

NOTA

- O DCS for Redis 3.0 não é mais fornecido. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.
- Não é possível atualizar a versão do Redis para uma instância. Por exemplo, uma instância do DCS Redis 3.0 de nó único não pode ser atualizada para uma instância do DCS Redis 4.0 ou 5.0 de nó único. Se o serviço exigir os recursos de versões superiores do Redis, crie uma instância do DCS Redis de uma versão superior e, em seguida, migre os dados da instância antiga para a nova.

Recursos

1. Baixa sobrecarga do sistema e alto QPS

   As instâncias de nó único não suportam sincronização de dados ou persistência de dados, reduzindo a sobrecarga do sistema e suportando maior simultaneidade. O QPS de instâncias do DCS Redis de nó único alcança até 100.000.

2. Monitoramento de processos e recuperação automática de falhas

   Com um mecanismo de monitoração HA, se uma instância de DCS do único-nó se torna defeituosa, um processo novo está começado dentro de 30 segundos para recomeçar o abastecimento de serviço.

3. Usabilidade out-of-the-box e sem persistência de dados

   As instâncias de DCS de nó único podem ser usadas fora da caixa porque não envolvem o carregamento de dados. Se o seu serviço exigir um alto QPS, você poderá aquecer os dados de antemão para evitar um forte impacto de simultaneidade no banco de dados de back-end.

4. Baixo custo e adequado para desenvolvimento e testes

   As instâncias de nó único são 40% mais baratas do que as instâncias de DCS principal/em espera, adequadas para configurar ambientes de desenvolvimento ou teste.

As instâncias de nó único suportam operações de leitura/gravação altamente simultâneas, mas não suportam a persistência de dados. Os dados serão excluídos depois que as instâncias

forem reiniciadas. Eles são adequados para cenários que não exigem persistência de dados, como cache de front-end de banco de dados, para acelerar o acesso e facilitar a carga de simultaneidade do backend. Se os dados desejados não existirem no cache, as solicitações irão para o banco de dados. Ao reiniciar o serviço ou a instância do DCS, você pode pré-gerar dados de cache do banco de dados do disco para aliviar a pressão sobre o back-end durante a inicialização.

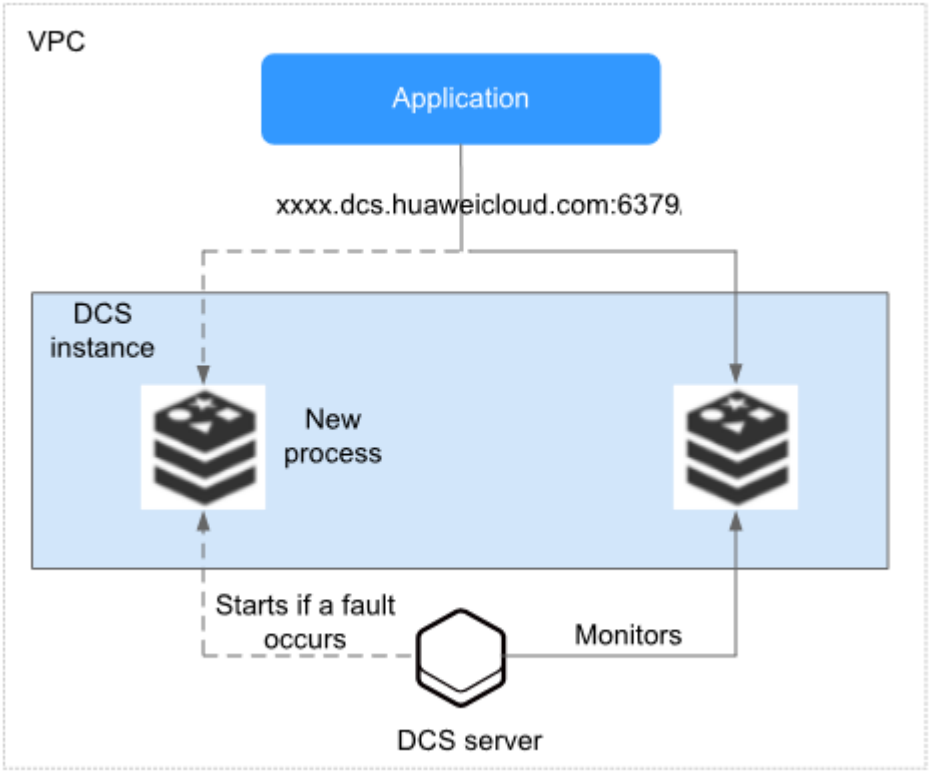
Arquitetura

Figura 3-1 mostra a arquitetura de instâncias do DCS Redis de nó único.

NOTA

O Redis 3.0 não suporta personalização de portas e permite apenas a porta 6379. Para Redis 4.0 e 5.0, você pode especificar uma porta ou usar a porta padrão 6379. Na arquitetura a seguir, a porta 6379 é usada. Se você personalizou uma porta, substitua **6379** pela porta real.

Figura 3-1 Arquitetura de instância do DCS Redis de nó único



Descrição da arquitetura

- **VPC**

  A VPC em que todos os nós da instância são executados.

NOTA

Para o acesso intra-VPC, o cliente e a instância devem estar na mesma VPC com configurações de regra de grupo de segurança especificadas.

Uma instância do DCS Redis 3.0 pode ser acessada de uma VPC ou em redes públicas. O cliente pode ser implantado fora da VPC e acessar a instância por meio do endereço IP elástico (EIP) vinculado à instância. O acesso público não é suportado pelas instâncias do DCS Redis 6.0, 4.0 e 5.0.

Para obter mais informações, consulte **Public a uma instância do DCS Redis** e **Como configuro um grupo de segurança?**

- **Application**

  O cliente da instância, que é o aplicativo em execução em um Elastic Cloud Server (ECS).

  As instâncias do DCS Redis e do Memcached são compatíveis com os protocolos Redis e Memcached, respectivamente, e podem ser acessadas por meio de clientes de código aberto. Para ver exemplos de acesso a instâncias DCS com diferentes linguagens de programação, consulte as **instruções de acesso a instâncias**.

- **DCS instance**

  Uma instância de DCS de nó único, que tem apenas um nó e um processo do Redis.

  O DCS monitora a disponibilidade da instância em tempo real. Se o processo do Redis ficar defeituoso, o DCS iniciará um novo processo em segundos para retomar o provisionamento de serviços.

3.2 Principal/em espera Redis

Tanto o DCS for Redis quanto o DCS for Memcached suportam o tipo de instância principal/em espera. Esta seção descreve instâncias do Memcached DCS principal/em espera. As versões do Redis disponíveis para instâncias principal/em espera do DCS Redis incluem o Redis 6.0, o Redis 5.0, o Redis 4.0 e o Redis 3.0.

A divisão de leitura/gravação é suportada por instâncias principal/em espera do DCS Redis 4.0 ou 5.0 por padrão, e não por instâncias principal/em espera do DCS Redis 3.0 e do Redis 6.0. Para obter detalhes, consulte **O DCS for Redis suporta divisão de leitura/gravação?**

NOTA

- O DCS for Redis 3.0 não é mais fornecido. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.
- Não é possível atualizar a versão do Redis para uma instância. Por exemplo, uma instância principal/em espera do DCS Redis 3.0 não pode ser atualizada para uma instância principal/em espera do DCS Redis 4.0 ou 5.0. Se o serviço exigir os recursos de versões superiores do Redis, crie uma instância do DCS Redis de uma versão superior e, em seguida, migre os dados da instância antiga para a nova.

Recursos

As instâncias principal/em espera têm maior disponibilidade e confiabilidade do que as instâncias de nó único.

As instâncias do Memcached DCS principal/em espera têm os seguintes recursos:

1. **Data persistence and high reliability**

Por padrão, a persistência de dados é ativada pelo nó principal e em espera de uma instância principal/em espera do Memcached DCS.

O nó de espera de uma instância do Redis 3.0 é invisível para você. Somente o nó principal fornece operações de leitura/gravação de dados.

O nó em espera de uma instância do Redis 4.0 ou 5.0 fica visível para você. Você pode ler dados do nó em espera conectando-se a ele usando o endereço somente leitura da instância.

O nó em espera de uma instância do Redis 6.0 fica visível para você. Você pode ler dados do nó em espera conectando-se a ele usando o endereço somente leitura da instância.

2. **Data synchronization**

   Os dados nos nós principal e em espera são mantidos consistentes por meio de sincronização incremental.

   NOTA

   Depois de se recuperar de uma exceção de rede ou falha de nó, as instâncias principal/em espera executam uma sincronização completa para garantir a consistência dos dados.

3. **Automatic principal/em espera switchover**

   Se o nó principal se tornar defeituoso, a instância será desconectada e indisponível por vários segundos. O nó em espera assume dentro de 30 segundos sem operações manuais para retomar os serviços estáveis.

4. **Multiple DR policies**

   Cada instância de DCS principal/em espera pode ser implantada em AZs com fontes de alimentação e redes fisicamente isoladas. Os aplicativos também podem ser implantados em AZs para obter HA para dados e aplicativos.

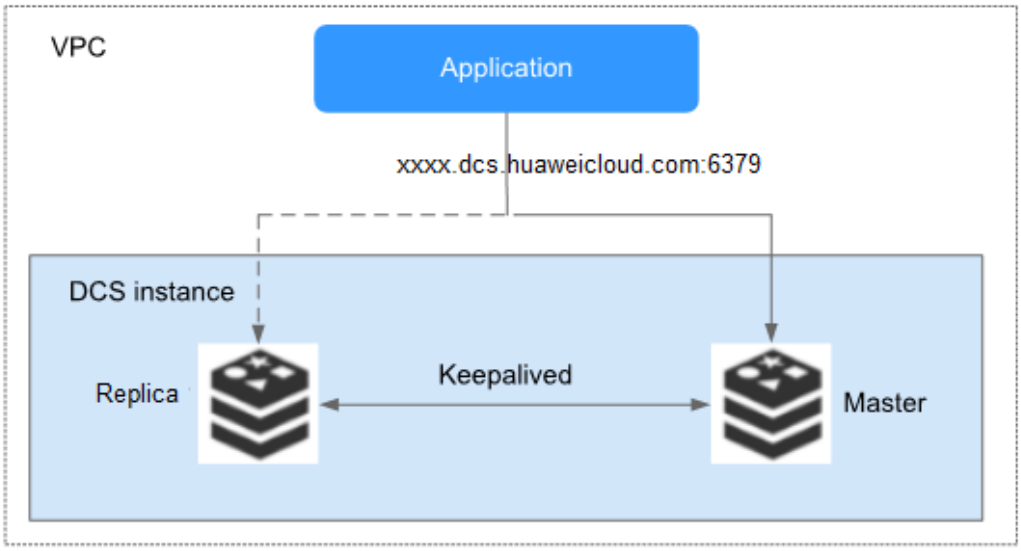
5. **Read/write splitting**

   As instâncias principal/em espera do DCS Redis 4.0 e 5.0 oferecem suporte à divisão de leitura/gravação do cliente. Ao se conectar a essa instância, você pode usar o endereço de leitura/gravação para se conectar ao nó principal ou usar o endereço somente leitura para se conectar ao nó em espera.

Arquitetura de instâncias principal/em espera do Redis 3.0

Figura 3-2 mostra a arquitetura de uma instância principal/em espera do DCS Redis 3.0.

Figura 3-2 Arquitetura de instância principal/em espera do DCS Redis 3.0



Descrição da arquitetura

- **VPC**

  A VPC em que todos os nós da instância são executados.

  NOTA

  Para o acesso intra-VPC, o cliente e a instância devem estar na mesma VPC com configurações de regra de grupo de segurança especificadas.

  Uma instância do DCS Redis 3.0 pode ser acessada de uma VPC ou em redes públicas. O cliente que acessa a instância pode ser implantado fora da VPC e acessar a instância por meio do EIP vinculado à instância. O acesso público não é suportado pelas instâncias do DCS Redis 6.0, 5.0 e 4.0.

  Para obter mais informações, consulte **Public a uma instância DCS Redis** e **Como configuro um grupo de segurança?**

- **Application**

  O cliente Memcached da instância, que é o aplicativo em execução no ECS.

  As instâncias do DCS Redis e do Memcached são compatíveis com os protocolos Redis e Memcached, respectivamente, e podem ser acessadas por meio de clientes de código aberto. Para obter exemplos de acesso a instâncias DCS com diferentes linguagens de programação, consulte as **instruções de acesso a instâncias**.

- **instância DCS**

  Uma instância de DCS principal/em espera que tem um nó principal e um nó de réplica. Por padrão, a persistência de dados é ativada e os dados são sincronizados entre os dois nós.

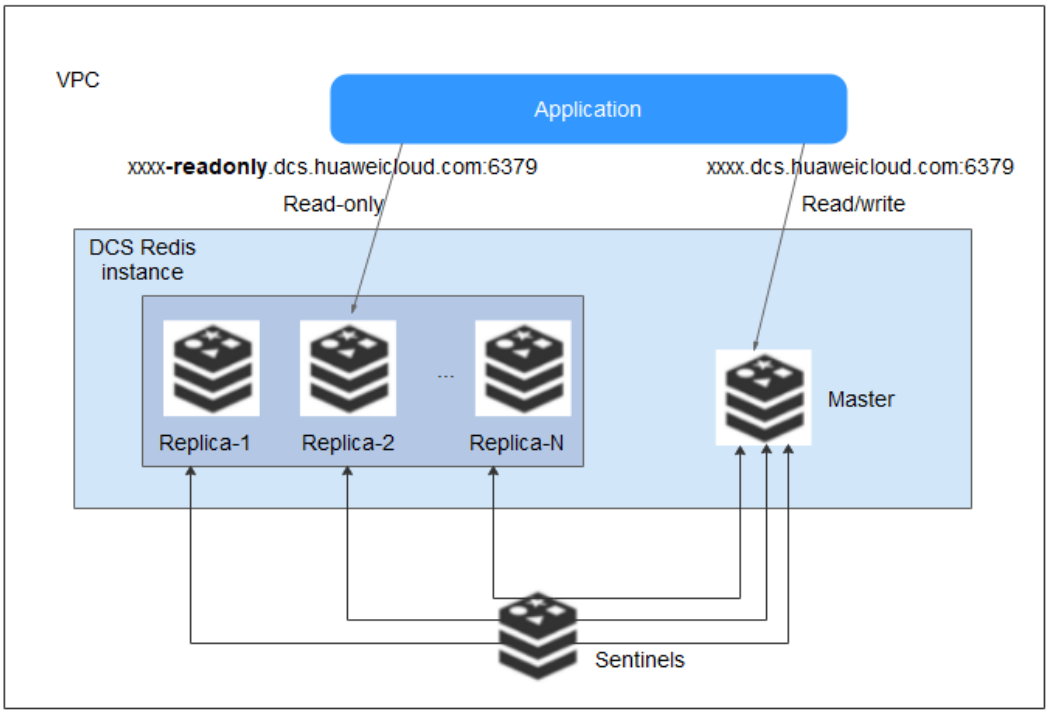
  O DCS monitora a disponibilidade da instância em tempo real. Se o nó principal se tornar defeituoso, o nó em espera se tornará o nó principal e retomará o provisionamento de serviços.

  A porta padrão do Redis é 6379.

Arquitetura de instâncias principal/em espera do Redis 4.0 e 5.0

Figura 3-3 A mostra a arquitetura das instâncias principal/em espera do DCS Redis 4.0 e 5.0.

Figura 3-3 Arquitetura de uma instância principal/em espera do DCS Redis 4.0 ou 5.0



Descrição da arquitetura

1. Cada instância principal/em espera do Redis 4.0 ou 5.0 tem dois endereços de conexão. Ao se conectar a essa instância, você pode usar o endereço de nome de domínio de leitura/gravação para se conectar ao nó principal ou usar o endereço de nome de domínio somente leitura para se conectar ao nó em espera.

   Os endereços de conexão podem ser obtidos na página de detalhes da instância no console do DCS.

2. Você pode configurar o Sentinel para uma instância principal/em espera do Redis 4.0 ou 5.0. Os sentinelas monitoram o status de execução dos nós principal e stand-by. Se o nó principal se tornar defeituoso, um failover será executado.

   As sentinelas são invisíveis para você e são usadas apenas no serviço. Para obter detalhes sobre o Sentinel, consulte **O que é o Sentinel?**

3. Um nó somente de leitura tem as mesmas especificações que um nó de leitura/gravação. Quando uma instância principal/em espera é criada, um par de nós principal e stand-by são incluídos na instância por padrão.

   NOTA

   - Para instâncias do DCS Redis 4.0 ou 5.0, você pode personalizar a porta. Se nenhuma porta for especificada, a porta padrão 6379 será usada. No diagrama de arquitetura, a porta 6379 é usada. Se você personalizou uma porta, substitua **6379** pela porta real.
   - Os nomes de domínio somente leitura das instâncias principal/em espera do DCS Redis 4.0 e 5.0 não oferecem suporte ao balanceamento de carga. Para alta confiabilidade e baixa latência, use cluster ou instâncias de divisão de leitura/gravação.

3.3 Cluster de proxy Redis

O DCS for Redis fornece dois tipos de instâncias de cluster: Cluster de proxy e cluster do Redis. O Cluster de Proxy é compatível com o Redis 3.0, 4.0 e 5.0 e usa o Linux Virtual Server (LVS) e proxies para alcançar alta disponibilidade. O Redis Cluster é a implementação distribuída nativa do Redis e é compatível com o Redis 4.0 e 5.0.

A divisão de leitura/gravação é suportada pela configuração do cliente para instâncias do Cluster Redis (Redis 4.0 e 5.0), mas não é suportada para instâncias do Cluster Proxy (Redis 3.0, 4.0 e 5.0). **Leia mais** sobre o suporte do DCS para divisão de leitura/gravação.

Esta seção descreve as instâncias do Cluster de Proxy DCS Redis 3.0, 4.0 e 5.0.

NOTA

- O DCS for Redis 3.0 não é mais fornecido. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.
- Não é possível atualizar a versão do Redis para uma instância. Por exemplo, uma instância do DCS Redis 3.0 de cluster de proxy não pode ser atualizada para uma instância do DCS Redis 4.0 ou 5.0 de cluster de proxy. Se o serviço exigir os recursos de versões superiores do Redis, crie uma instância do DCS Redis de uma versão superior e, em seguida, migre os dados da instância antiga para a nova.
- Uma instância de Cluster de Proxy pode ser conectada da mesma maneira que uma instância de nó único ou principal/em espera, sem nenhuma configuração especial no cliente. Você pode usar o endereço IP ou o nome de domínio da instância e não precisa saber ou usar os endereços proxy ou shard.

Instâncias do Cluster Proxy DCS Redis 3.0

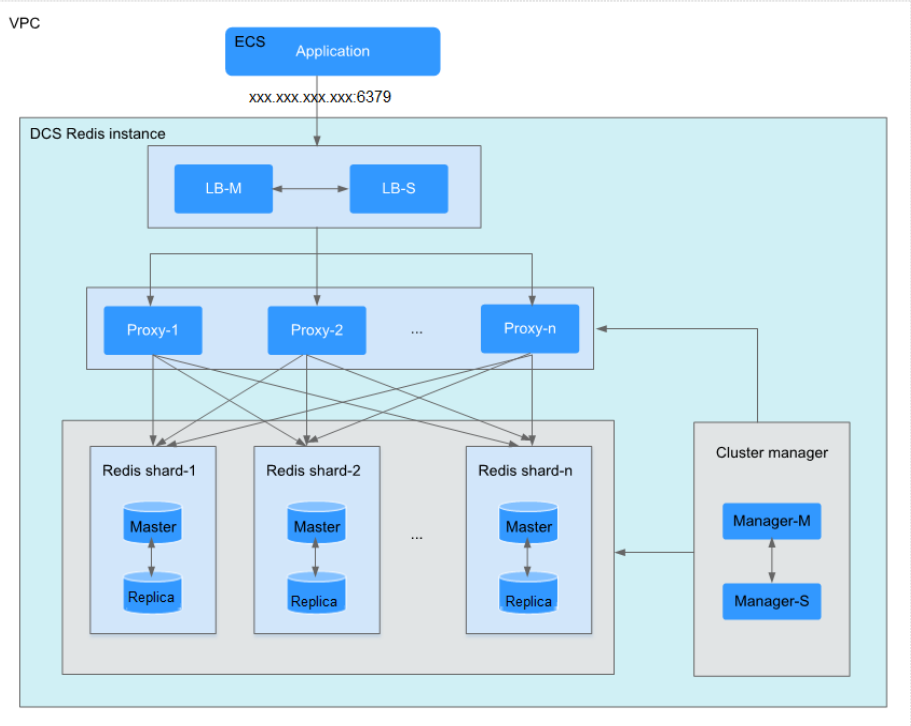
As instâncias do DCS Redis 3.0 são baseadas em x86, compatíveis com **códigos de código aberto** e vêm com especificações que variam de 64 GB a 1024 GB, atendendo aos requisitos de **millions of concurrent connections** e **massive data cache**. O armazenamento de dados distribuídos e o acesso são implementados pela DCS, sem necessidade de desenvolvimento ou manutenção.

Cada instância de cluster de proxy consiste em balanceadores de carga, proxies, gerenciadores de cluster e **estilhaços**.

Tabela 3-1 Especificações das instâncias do cluster de proxy DCS Redis 3.0

Memória Total	Proxies	Partições
64 GB	3	8
128 GB	6	16
256 GB	8	32
512 GB	16	64
1024 GB	32	128

Figura 3-4 Arquitetura de uma instância do DCS Redis 3.0 de cluster de proxy



Descrição da arquitetura

- **VPC**

  A VPC em que todos os nós da instância são executados.

  NOTA

  Se o acesso público não estiver habilitado para a instância, verifique se o cliente e a instância estão na mesma VPC e configure regras de grupo de segurança para a VPC.

  Se o acesso público estiver habilitado para a instância, o cliente poderá ser implantado fora da VPC para acessar a instância por meio do EIP vinculado à instância.

  Para obter mais informações, consulte **Public a uma instância DCS Redis 3.0** e **Como configuro um grupo de segurança?**

- **Application**

  O cliente usado para acessar a instância.

  As instâncias do DCS Redis podem ser acessadas usando clientes de código aberto. Para ver exemplos de acesso a instâncias de DCS com diferentes linguagens de programação, consulte **Accessing uma instância de DCS Redis**.

- **LB-M/LB-S**

  Os balanceadores de carga, que são implantados no modo HA principal/em espera. Os endereços de conexão (**IP address:Port** e **Domain Name:Port**) da instância do DCS Redis do cluster são os endereços dos balanceadores de carga.

- **Proxy**

  O servidor proxy usado para alcançar alta disponibilidade e processar solicitações de clientes de alta concorrência.

  Você pode se conectar a uma instância de Cluster de Proxy nos endereços IP de seus proxies.

- **Redis shard**

  Um fragmento do cluster.

  Cada estilhaço consiste em um par de nós principal/réplica. Se o nó principal se tornar defeituoso, o nó de réplica assume automaticamente os serviços de cluster.

  Se ambos os nós principal e réplica de um estilhaço estiverem com defeito, o cluster ainda pode fornecer serviços, mas os dados no estilhaço com defeito estão inacessíveis.

- **Cluster manager**

  Os gerenciadores de configuração de cluster, que armazenam configurações e políticas de particionamento do cluster. Não é possível modificar as informações sobre os gerenciadores de configuração.

Instâncias do Cluster Proxy DCS Redis 4.0 e 5.0

NOTA

As instâncias de cluster de proxy DCS Redis 4.0 e 5.0 são fornecidas apenas em algumas regiões.

As instâncias do Redis 4.0 e 5.0 são criadas com base no Redis 4.0 e 5.0 de código aberto e compatíveis com **o Codis de código aberto**. Eles fornecem várias especificações de grande capacidade que variam de 4 GB a 1024 GB e suportam as arquiteturas x86 e Arm CPU.

Tabela 3-2 lista o número de estilhaços correspondentes a especificações diferentes. Você pode personalizar o tamanho do estilhaço ao criar uma instância. Atualmente, o número de estilhaços e réplicas não pode ser personalizado. Por padrão, cada estilhaço tem duas réplicas.

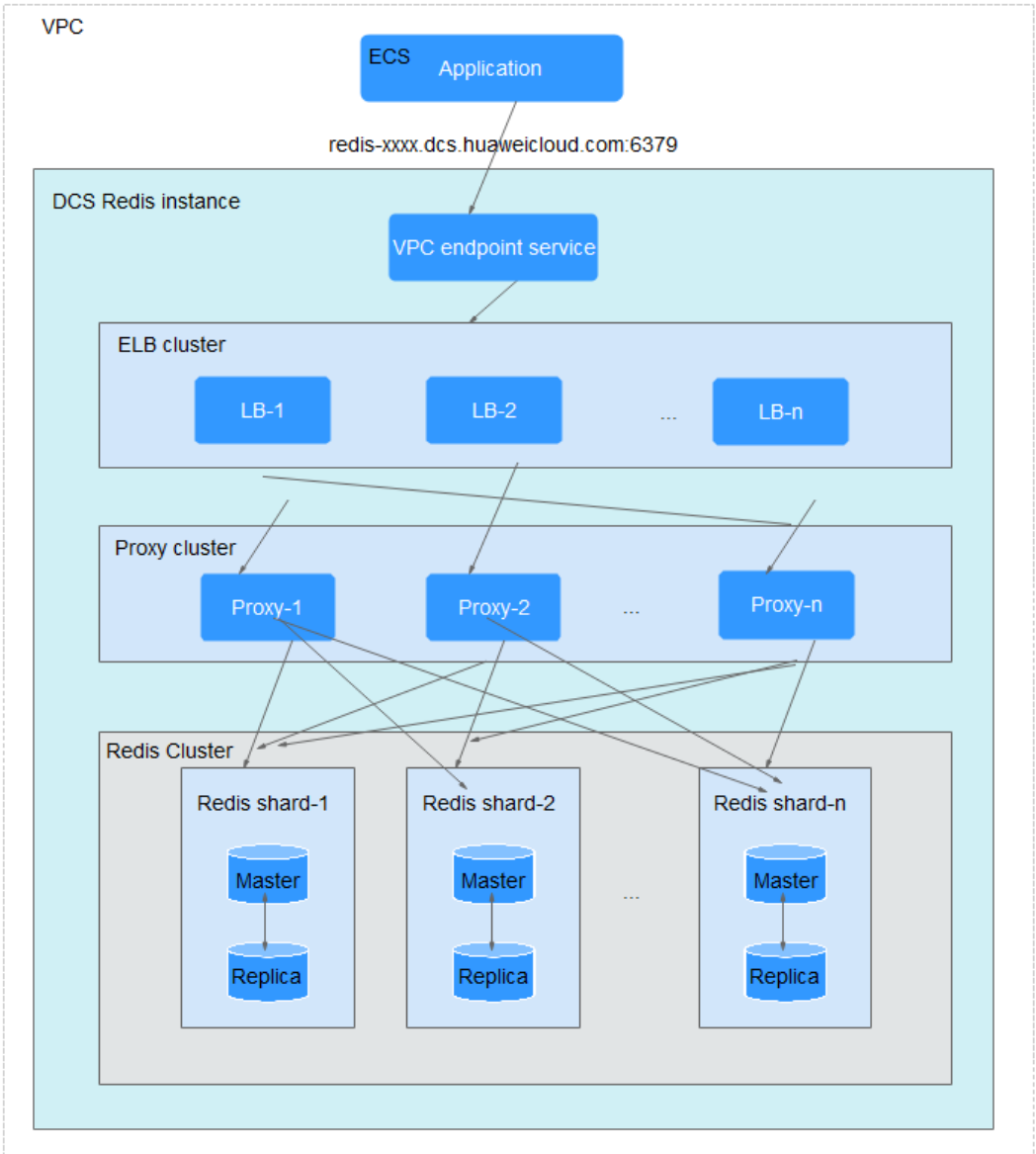
Memory per shard=Instance specification/Number of shards. Por exemplo, se uma instância de 48 GB tiver 6 estilhaços, o tamanho de cada estilhaço será 48 GB/6 = 8 GB.

Tabela 3-2 Especificações das instâncias de cluster de proxy DCS Redis 4.0 e 5.0

Memória Total	Proxies	Partições	Memória por Fragmento (GB)
4 GB	3	3	1,33
8 GB	3	3	2,67
16 GB	3	3	5,33
24 GB	3	3	8
32 GB	3	3	10.67
48 GB	6	6	8
64 GB	8	8	8
96 GB	12	12	8
128 GB	16	16	8
192 GB	24	24	8
256 GB	32	32	8
384 GB	48	48	8

Memória Total	Proxies	Partições	Memória por Fragmento (GB)
512 GB	64	64	8
768 GB	96	96	8
1024 GB	128	128	8

Figura 3-5 Arquitetura de um cluster de proxy Instância do DCS Redis 4.0 ou 5.0



Descrição da arquitetura

- **VPC**

  A VPC em que todos os nós da instância são executados.

> NOTA
>
> O cliente e a instância do cluster devem estar na mesma VPC, e a lista de permissões da instância deve permitir o acesso do endereço IP do cliente.

- **Application**

  O cliente usado para acessar a instância.

  As instâncias do DCS Redis podem ser acessadas usando clientes de código aberto. Para ver exemplos de acesso a instâncias de DCS com diferentes linguagens de programação, consulte **Accessing uma instância de DCS Redis**.

- **VPC endpoint service**

  Você pode configurar sua instância do DCS Redis como um serviço de endpoint da VPC e acessar a instância usando o endereço do serviço de endpoint da VPC.

  O endereço IP ou o endereço do nome de domínio da instância do DCS Redis de cluster de proxy é o endereço do serviço de endpoint da VPC.

- **ELB**

  Os balanceadores de carga são implantados no modo HA de cluster e oferecem suporte à implantação multi-AZ.

- **Proxy**

  O servidor proxy usado para obter alta disponibilidade e processar solicitações de clientes de alta concorrência.

  Você não pode se conectar a uma instância de Cluster de Proxy nos endereços IP de seus proxies.

- **Redis Cluster**

  Um fragmento do aglomerado.

  Cada estilhaço consiste em um par de nós principal/réplica. Se o nó principal se tornar defeituoso, o nó de réplica assume automaticamente os serviços de cluster.

  Se os nós principal e réplica de um estilhaço estiverem defeituosos, o cluster ainda poderá fornecer serviços, mas os dados no estilhaço defeituoso estarão inacessíveis.

3.4 Cluster do Redis

O DCS fornece dois tipos de instâncias do Redis de cluster: Cluster de proxy e cluster do Redis. O cluster de proxy usa o Linux Virtual Server (LVS) e proxies. O Redis Cluster é a implementação distribuída nativa do Redis. As instâncias de cluster proxy são compatíveis com o Redis 3.0, 4.0 e 5.0, enquanto as instâncias de cluster do Redis são compatíveis com o Redis 4.0 e 5.0.

A divisão de leitura/gravação é suportada pela configuração do cliente para instâncias do Cluster Redis (Redis 4.0 e 5.0), mas não é suportada para instâncias do Cluster Proxy (Redis 3.0, 4.0 e 5.0). **Leia mais** sobre o suporte do DCS para divisão de leitura/gravação.

Esta seção descreve as instâncias do Cluster de Proxy DCS Redis 3.0, 4.0 e 5.0.

Instâncias do Cluster Proxy DCS Redis 4.0 e 5.0

O tipo de instância do Cluster do Redis fornecido pelo DCS é compatível com o **Cluster do Redis nativo**, que usa clientes inteligentes e uma arquitetura distribuída para executar sharding.

Tabela 3-3 lista as especificações de estilhaços para diferentes especificações de instância.

Você pode personalizar o tamanho do estilhaço ao criar uma instância do Cluster do Redis. Se o tamanho do estilhaço não for personalizado, o tamanho padrão será usado. **Size of a shard = Instance specification/Number of shards**. Por exemplo, se uma instância de 48 GB tiver 6 estilhaços, o tamanho de cada estilhaço será 48 GB/6 = 8 GB.

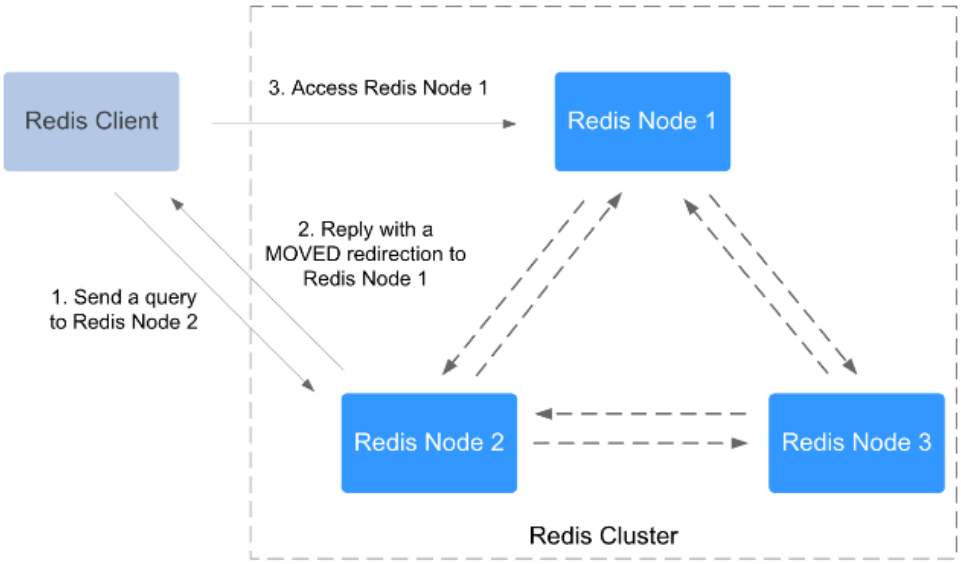
Tabela 3-3 Especificações das instâncias de DCS do Cluster do Redis

Memória Total	Partições
4 GB/8 GB/16 GB/24 GB/32 GB	3
48 GB	6
64 GB	8
96 GB	12
128 GB	16
192 GB	24
256 GB	32
384 GB	48
512 GB	64
768 GB	96
1024 GB	128

- Arquitetura distribuída

  Qualquer nó em um cluster do Redis pode receber solicitações. As solicitações recebidas são então redirecionadas para o nó certo para processamento. Cada nó consiste em um subconjunto de um principal e uma (por padrão) ou várias réplicas. As funções de principal ou réplica são determinadas por meio de um algoritmo de eleição.

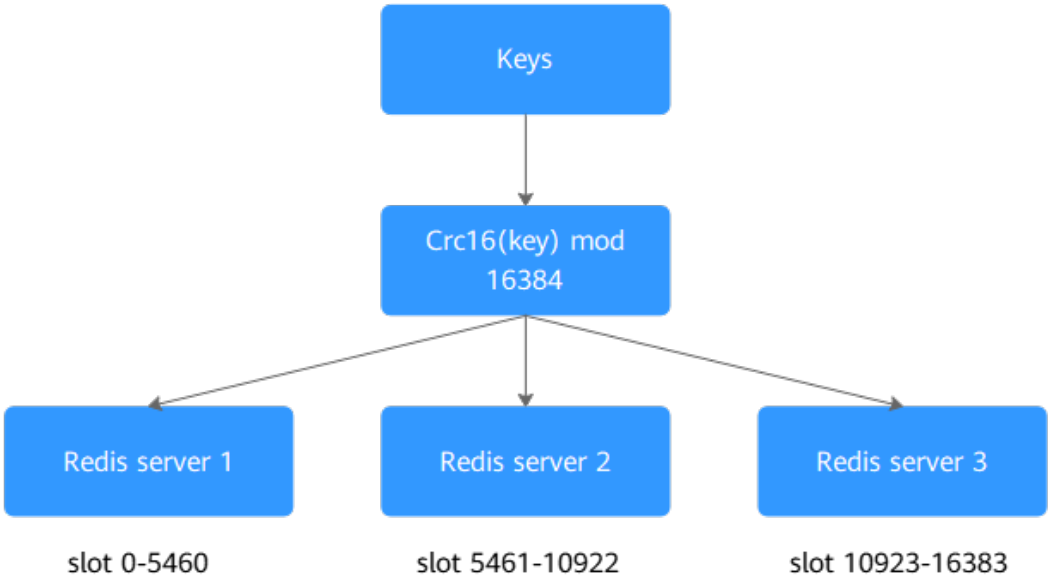
Figura 3-6 Arquitetura distribuída do Redis Cluster



- Presharding

  Há slots de hash 16.384 em cada cluster do Redis. O mapeamento entre slots de hash e nós do Redis é armazenado em servidores Redis. Para calcular qual é o slot de hash de uma determinada chave, basta pegar o CRC16 da chave módulo 16384.

Figura 3-7 Pré-harding do cluster do Redis



3.5 Separação de leitura/gravação

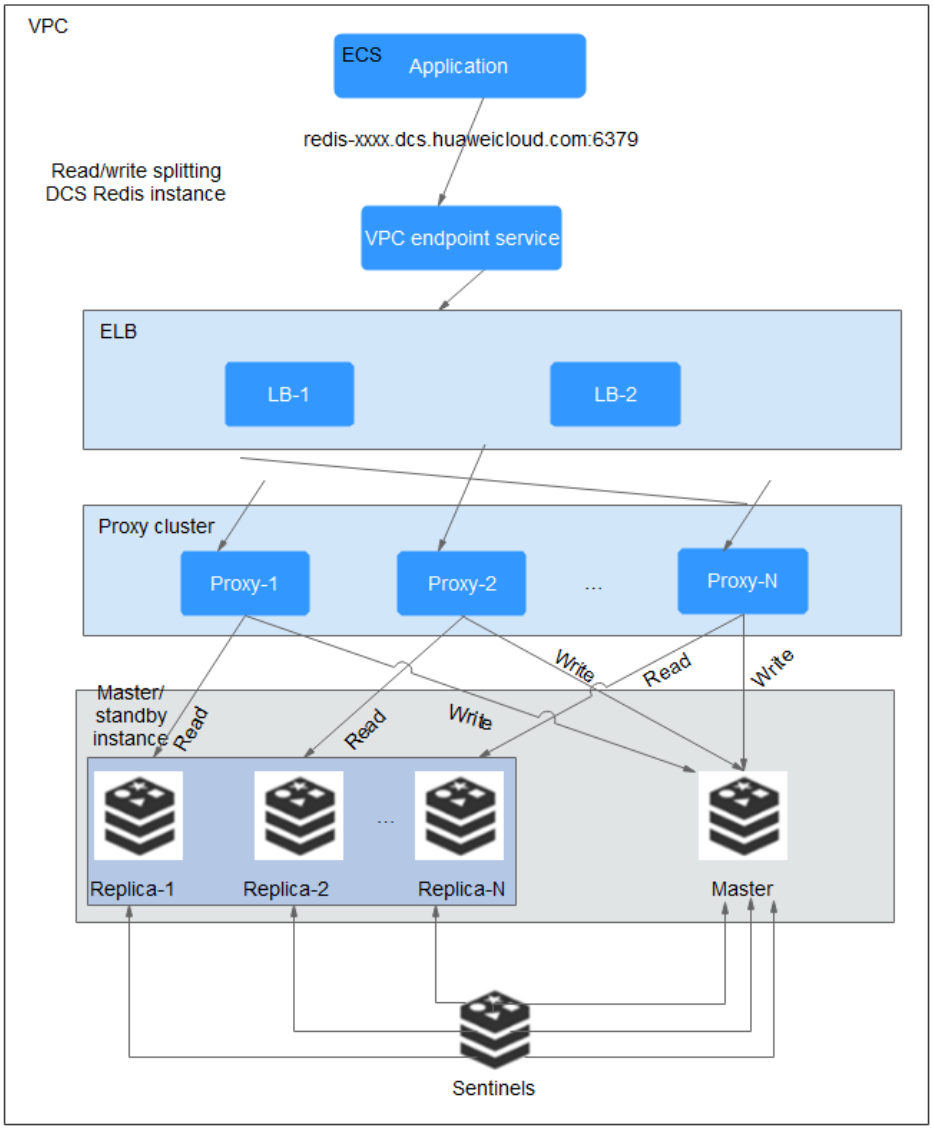
Esta seção descreve as instâncias de DCS principal/em espera nas quais a divisão de leitura/gravação é implementada no lado do servidor ou do lado do cliente. A divisão de leitura/gravação é adequada para cenários com alta simultaneidade de leitura e poucas solicitações de gravação, com o objetivo de melhorar o desempenho de alta simultaneidade e reduzir os custos de O&M.

> **NOTA**
>
> A divisão de leitura/gravação do servidor e a divisão de leitura/gravação do cliente são suportadas apenas em algumas regiões.

- Para uma instância de divisão de leitura/gravação do DCS Redis 4.0 ou 5.0, a divisão de leitura/gravação é implementada no lado do servidor por padrão. Os proxies distinguem entre solicitações de leitura e solicitações de gravação e encaminham solicitações de gravação para o nó principal e solicitações de leitura para o nó em espera. Não é necessário configurá-los novamente.
- Para uma instância principal/em espera do DCS Redis 4.0 ou 5.0, a divisão de leitura/gravação é implementada no lado do cliente por padrão. A instância fornece um nome de domínio de leitura/gravação e um nome de domínio somente leitura para receber, respectivamente, solicitações de leitura e gravação do cliente. Desta forma, a divisão de leitura/gravação é alcançada.

Divisão de leitura/gravação do servidor

Figura 3-8 Arquitetura de uma instância de divisão de leitura/gravação



Descrição da arquitetura

- **VPC endpoint service**

  Você pode configurar sua instância do DCS Redis como um serviço de endpoint da VPC e acessar a instância usando o endereço do serviço de endpoint da VPC.

  O endereço IP ou o nome de domínio da instância do DCS Redis de divisão de leitura/gravação é o endereço do serviço de endpoint da VPC.

- **ELB**

  Os balanceadores de carga são implantados no modo HA de cluster e oferecem suporte à implantação multi-AZ.

- **Proxy**

  Um cluster de proxy é usado para distinguir entre solicitações de leitura e solicitações de gravação, e encaminhar solicitações de gravação para o nó principal e solicitações de leitura para o nó em espera. Não é necessário configurá-los novamente.

- **Sentinel cluster**

  Sentinelas monitoram o status do principal e réplicas. Se o nó principal estiver com defeito ou anormal, um failover é executado para garantir que os serviços não sejam interrompidos.

- **Master/standby instance**

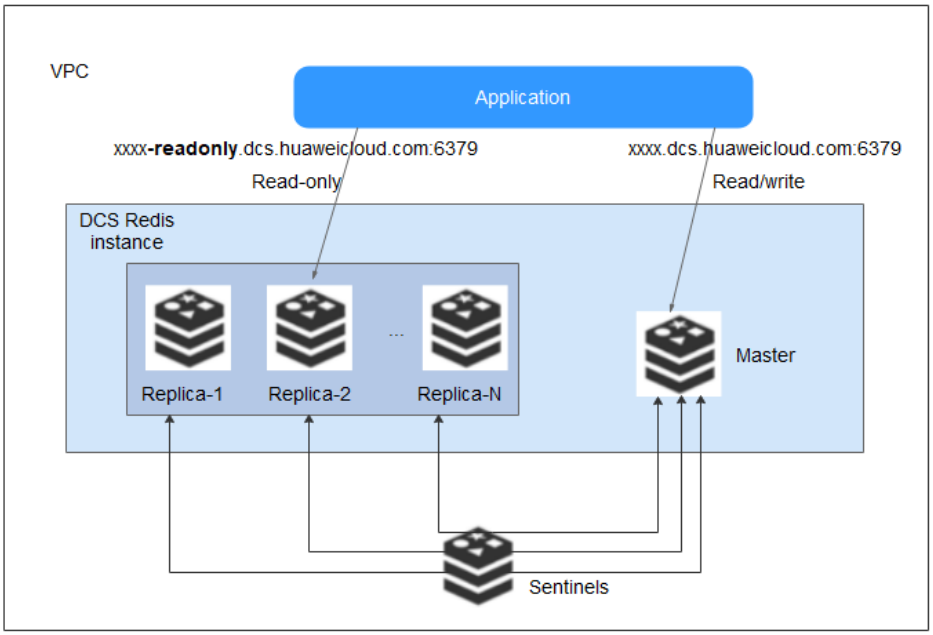
  Uma instância de divisão de leitura/gravação é essencialmente uma instância principal/em espera que consiste em um nó principal e um nó em espera. Por padrão, a persistência de dados é ativada e os dados são sincronizados entre os dois nós.

  Os nós principal e em espera podem ser implantados em diferentes AZs.

Divisão de leitura/gravação do cliente

Figura 3-9 A mostra a arquitetura de uma instância principal/em espera do DCS Redis 4.0 e 5.0.

Figura 3-9 Arquitetura de uma instância principal/em espera do DCS Redis 4.0 ou 5.0



Descrição da arquitetura:

1. Uma instância principal/em espera de divisão de leitura/gravação do DCS Redis 4.0 ou 5.0 tem um nome de domínio de leitura/gravação e um nome de domínio somente leitura. O cliente distingue entre solicitações de leitura e solicitações de gravação e envia solicitações de gravação para o nome de domínio de leitura/gravação e solicitações de leitura para o nome de domínio somente leitura para se conectar ao nó principal ou ao nó em espera.

   Os dois nomes de domínio podem ser obtidos na página de detalhes da instância no console.
2. As instâncias principal/em espera do DCS Redis 4.0 e 5.0 oferecem suporte a Sentinels. Os sentinelas monitoram o status de execução dos nós principal e em espera. Se o nó principal se tornar defeituoso, um failover será executado.

   As sentinelas são invisíveis para você e são usadas apenas no serviço.
3. Um nó somente de leitura tem as mesmas especificações que um nó de leitura/gravação. Uma instância principal/em espera consiste em um par de nós principal e em espera por padrão.

Ao usar instâncias de divisão de leitura/gravação, observe o seguinte:

1. As solicitações de leitura são enviadas às réplicas. Há um atraso quando os dados são sincronizados do principal para as réplicas.

   Se seus serviços forem sensíveis ao atraso, não use instâncias de divisão de leitura/gravação. Em vez disso, use instâncias principal/em espera ou cluster.
2. A divisão de leitura/gravação é adequada quando há mais solicitações de leitura do que solicitações de gravação. Se houver muitas solicitações de gravação, o principal e as réplicas podem ser desconectadas ou a sincronização de dados entre elas pode falhar após a desconexão. Como resultado, o desempenho de leitura se deteriora.

   Se os seus serviços forem pesados em gravação, use instâncias principal/em espera ou de cluster.
3. Se uma réplica estiver com defeito, leva algum tempo para sincronizar todos os dados do principal. Durante a sincronização, a réplica não fornece serviços e o desempenho de leitura da instância se deteriora.

   Para reduzir o impacto da falha, use uma instância com menos de 32 GB de memória. Quanto menor a memória, menor o tempo para sincronização completa de dados entre o principal e as réplicas, e menor o impacto da interrupção.

3.6 Comparando Tipos de Instância do DCS Redis

Tabela 3-4 descreve as diferenças entre os diferentes tipos de instância do Redis em termos de recursos e comandos.

NOTA

O DCS for Redis 3.0 não é mais fornecido. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.

Tabela 3-4 Diferenças entre os tipos de instância de DCS

Item	Nó único ou principal/em espera	Cluster de proxy	Cluster do Redis
Compatibilidade da versão do Redis	Redis 3.0, 4.0 e 5.0 O Redis 6.0 é compatível com o KeyDB Open Source (disponível apenas para instâncias principal/em espera). Você pode selecionar uma versão ao criar uma instância.	Redis 3.0, 4.0 e 5.0	Redis 4.0 e 5.0 Você pode selecionar uma versão ao criar uma instância.
Apoio	● Notificações do keyspace ● Pipelinização	● Pipelining, comando **MSET** e comando **MGET** ● Comando **SCAN**, comando **KEYS** e registro lento do Redis ● Pub/Sub	● Notificações do keyspace ● Comandos **BRPOP**, **BLPOP** e **BRPOPLPUSH** ● Pub/Sub

Item	Nó único ou principal/em espera	Cluster de proxy	Cluster do Redis
Restrições	A persistência de dados não é suportada para instâncias de nó único.	● O script LUA é restrito: Todas as chaves devem estar no mesmo slot de hash para evitar erros. Hash tags são recomendadas. ● Se um comando contiver várias chaves, as chaves devem estar no mesmo hash slot para evitar erros. Hash tags são recomendadas. ● As notificações de keyspace não são suportadas.	● O script LUA é restrito: Todas as chaves devem estar no mesmo hash slot. Hash tags são recomendadas. ● O SDK cliente deve oferecer suporte ao Redis Cluster e ser capaz de processar erros MOVED. ● Quando você estiver usando pipelining, comando **MSET** ou comando **MGET**, todas as chaves devem estar no mesmo slot de hash para evitar erros. Hash tags são recomendadas. ● Ao usar notificações de keyspace, estabeleça conexões com cada servidor Redis e processe eventos em cada conexão. ● Ao usar um comando de deslocamento ou global, como **SCAN** e **KEYS**, execute o comando em cada servidor Redis.
Cliente	Qualquer cliente Redis	Qualquer cliente Redis (não é necessário oferecer suporte ao protocolo Redis Cluster)	Qualquer cliente que suporte o protocolo Redis Cluster
Comandos desativados	Alguns comandos do Redis não são suportados. Para obter detalhes, consulte **Tabela 5-3**, **Tabela 5-11**, e **Tabela 5-21**.	Alguns comandos do Redis não são suportados. Para mais detalhes, consulte **Tabela 5-4**.	Alguns comandos do Redis não são suportados. Para obter mais detalhes, consulte **Tabela 5-13** e **Tabela 5-23**.

Item	Nó único ou principal/em espera	Cluster de proxy	Cluster do Redis
Réplicas	Uma instância de nó único tem apenas uma réplica. Uma instância principal/em espera tem duas réplicas. Atualmente, o número de réplicas não pode ser personalizado para instâncias principal/em espera do DCS Redis 3.0 e do DCS Redis 6.0. Por padrão, uma instância principal/em espera tem um nó principal e um nó stand-by. Ao criar uma instância principal/em espera do DCS Redis 4.0 ou 5.0, você pode personalizar o número de réplicas, sendo uma delas a principal.	Cada fragmento no cluster tem e só pode ter duas réplicas, sendo uma delas a principal.	Por padrão, cada estilhaço no cluster tem duas réplicas. O número de réplicas pode ser personalizado, com um deles sendo o principal. Ao criar uma instância, você pode definir a quantidade de réplicas como uma, indicando que a instância tem apenas o nó principal. Neste caso, não é possível garantir uma elevada fiabilidade dos dados.

3.7 Memcached de nó único (indisponível em breve)

NOTA

O DCS for Memcached está prestes a ficar indisponível e não é mais vendido em algumas regiões. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.

Esta seção descreve os recursos e a arquitetura de instâncias do Memcached DCS de nó único.

Recursos

1. Baixa sobrecarga do sistema e alto QPS

   As instâncias de nó único não suportam sincronização de dados ou persistência de dados, reduzindo a sobrecarga do sistema e suportando maior simultaneidade. O QPS de instâncias do DCS Redis de nó único alcança até 100.000.

2. Monitoramento de processos e recuperação automática de falhas

   Com um mecanismo de monitoração HA, se uma instância de DCS do único-nó se torna defeituosa, um processo novo está começado dentro de 30 segundos para recomeçar o abastecimento de serviço.

3. Usabilidade out-of-the-box e sem persistência de dados

   As instâncias de DCS de nó único podem ser usadas fora da caixa porque não envolvem o carregamento de dados. Se o seu serviço exigir um alto QPS, você poderá aquecer os dados de antemão para evitar um forte impacto de simultaneidade no banco de dados de back-end.

4. Baixo custo e adequado para desenvolvimento e testes

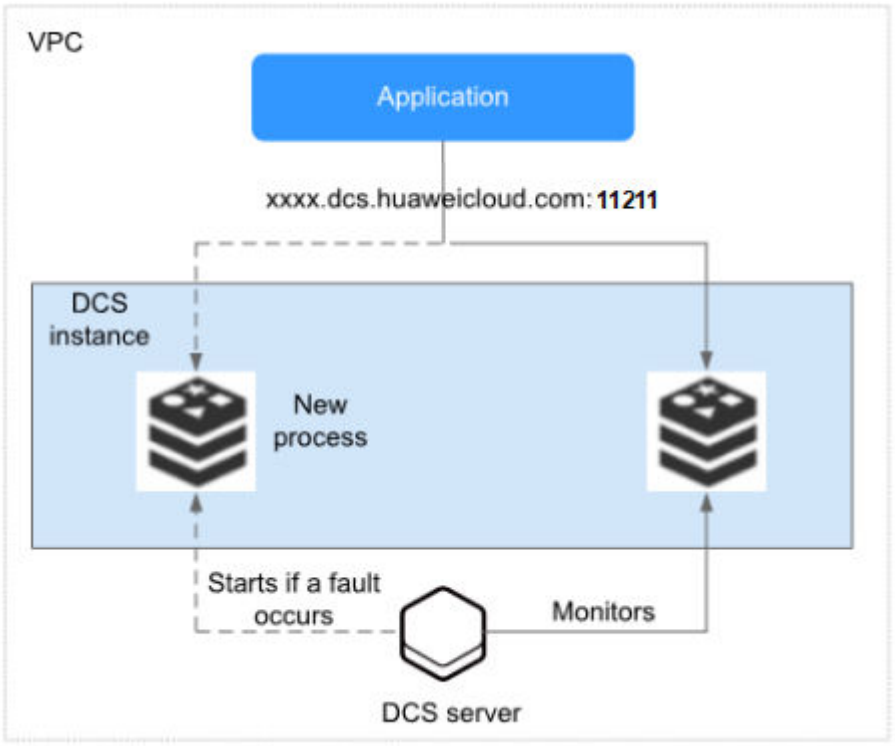
   As instâncias de nó único são 40% mais baratas do que as instâncias de DCS principal/em espera, adequadas para configurar ambientes de desenvolvimento ou teste.

As instâncias de nó único suportam operações de leitura/gravação altamente simultâneas, mas não suportam a persistência de dados. Os dados serão excluídos depois que as instâncias forem reiniciadas. Eles são adequados para cenários que não exigem persistência de dados, como cache de front-end de banco de dados, para acelerar o acesso e facilitar a carga de simultaneidade do backend. Se os dados desejados não existirem no cache, as solicitações irão para o banco de dados. Ao reiniciar o serviço ou a instância do DCS, você pode pré-gerar dados de cache do banco de dados do disco para aliviar a pressão sobre o back-end durante a inicialização.

Arquitetura

Figura 3-10 mostra a arquitetura de instâncias do Memcached DCS de nó único.

Figura 3-10 Arquitetura de instância do DCS Redis de nó único



Descrição da arquitetura

- **VPC**

  A VPC em que todos os nós da instância são executados.

  NOTA

  As instâncias do Memcached DCS de nó único não oferecem suporte ao acesso público. O cliente e a instância devem estar na mesma VPC com configurações de regra de grupo de segurança.

  Para obter mais informações, consulte **Como configuro um grupo de segurança?**

- **Application**

  O cliente da instância, que é o aplicativo em execução em um Elastic Cloud Server (ECS).

  As instâncias do Memcached do DCS são compatíveis com o protocolo Memcached e podem ser acessadas por meio de clientes de código aberto. Para ver exemplos de acesso a instâncias de DCS com diferentes linguagens de programação, consulte **Acessando uma instância de DCS Redis**.

- **DCS instance**

  Uma instância de DCS de nó único, que tem apenas um nó e um processo do Redis.

  O DCS monitora a disponibilidade da instância em tempo real. Se o processo do Memcached se tornar defeituoso, o DCS iniciará um novo processo para retomar o provisionamento de serviços.

  Use a porta 11211 para acessar uma instância do Memcached DCS.

3.8 Memcached principal/em espera (indisponível em breve)

NOTA

O DCS for Memcached está prestes a ficar indisponível e não é mais vendido em algumas regiões. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.

Esta seção descreve instâncias do Memcached DCS principal/em espera.

Recursos

As instâncias principal/em espera têm maior disponibilidade e confiabilidade do que as instâncias de nó único.

As instâncias do Memcached DCS principal/em espera têm os seguintes recursos:

1. **Data persistence and high reliability**

   Por padrão, a persistência de dados é ativada pelo nó principal e stand-by de uma instância principal/em espera do Memcached DCS. Além disso, a persistência de dados é suportada para garantir alta confiabilidade dos dados.

   O nó stand-by de uma instância do Memcached DCS é invisível para você. Somente o nó principal fornece operações de leitura/gravação de dados.

2. **Data synchronization**

   Os dados nos nós principal e stand-by são mantidos consistentes por meio de sincronização incremental.

   NOTA

   Depois de se recuperar de uma exceção de rede ou falha de nó, as instâncias principal/em espera executam uma sincronização completa para garantir a consistência dos dados.

3. **Comutação automática de mestre/standby**

   Se o nó principal se tornar defeituoso, o nó de espera assume dentro de 30 segundos, sem exigir interrupções de serviço ou operações manuais.

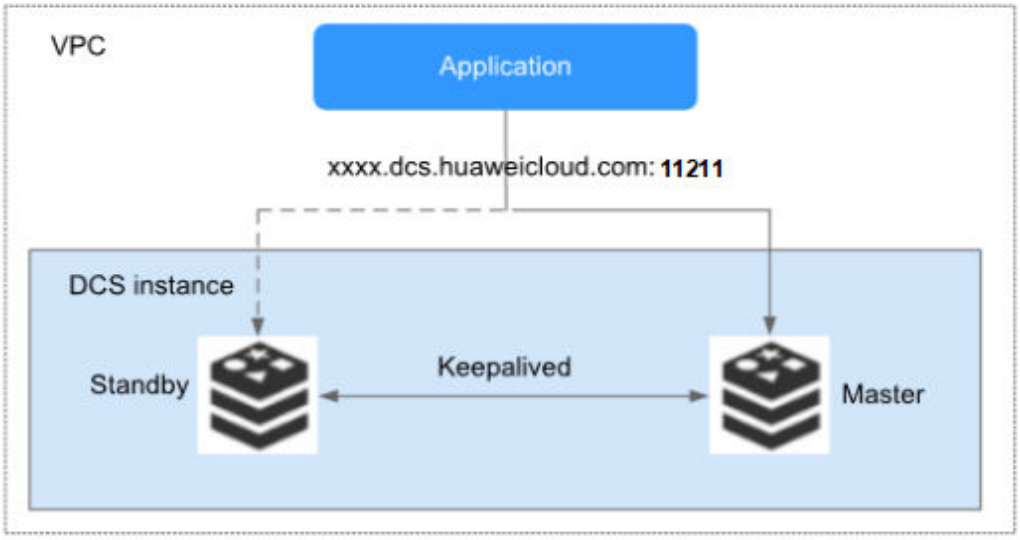
4. **Multiple DR policies**

   Cada instância de DCS principal/em espera pode ser implantada em AZs com fontes de alimentação e redes fisicamente isoladas. Os aplicativos também podem ser implantados em AZs para obter HA para dados e aplicativos.

Arquitetura de instâncias principal/em espera do DCS Redis 6.0

Figura 3-11 mostra a arquitetura de instâncias principal/em espera do Memcached DCS.

Figura 3-11 Arquitetura de instância do Memcached DCS principal/em espera



Descrição da arquitetura

- **VPC**

  A VPC em que todos os nós da instância são executados.

  NOTA

  As instâncias do Memcached DCS principal/em espera não oferecem suporte ao acesso público. O cliente e a instância devem estar na mesma VPC com configurações de regra de grupo de segurança.

  Para obter mais informações, consulte **Como configuro um grupo de segurança?**

- **Application**

  O cliente Memcached da instância, que é o aplicativo em execução no ECS.

  As instâncias do Memcached do DCS são compatíveis com o protocolo Memcached e podem ser acessadas por meio de clientes de código aberto. Para ver exemplos de acesso a instâncias de DCS com diferentes linguagens de programação, consulte **Acessando uma instância de DCS Redis**.

- **DCS instance**

  Indica uma instância de DCS principal/em espera que tem um nó principal e um nó stand-by. Por padrão, a persistência de dados é ativada e os dados são sincronizados entre os dois nós.

  O DCS monitora a disponibilidade da instância em tempo real. Se o nó principal se tornar defeituoso, o nó em espera se tornará o nó principal e retomará o provisionamento de serviços.

  Use a porta 11211 para acessar uma instância do Memcached DCS.

4 Especificações da instância de DCS

4.1 Especificações de instância do Redis 3.0 (descontinuado)

Esta seção descreve as especificações da instância do DCS Redis 3.0, incluindo a memória total, a memória disponível, o número máximo de conexões permitidas, a largura de banda máxima/garantida e o desempenho de referência.

As métricas a seguir estão relacionadas às especificações da instância:

- Memória usada: Você pode verificar o uso de memória de uma instância visualizando as métricas **Memory Usage** e **Used Memory**.
- Máximo de conexões: O número máximo de conexões permitidas é o número máximo de clientes que podem ser conectados a uma instância. Para verificar o número de conexões com uma instância, exiba a métrica **Connected Clients**.
- O QPS representa consultas por segundo, que é o número de comandos processados por segundo.

NOTA

- As instâncias do DCS Redis 3.0 estão disponíveis nos tipos de cluster de nó único, principal/em espera e proxy.
- A venda da DCS for o Redis 3.0 foi descontinuada. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.

Instâncias de nó único

Para cada instância do DCS Redis de nó único, a memória disponível é menor do que a memória total porque alguma memória é reservada para sobrecargas do sistema, conforme mostrado na **Tabela 4-1**.

Tabela 4-1 Especificações de instâncias do DCS Redis 3.0 de nó único

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/Limite) (Contagem)	Largura de banda assegurada/máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
2	1,5	5000/50.000	42/512	50.000	dcs.single_node
4	3.2	5000/50.000	64/1536	100.000	dcs.single_node
8	6,8	5000/50.000	64/1536	100.000	dcs.single_node
16	13,6	5000/50.000	85/3072	100.000	dcs.single_node
32	27,2	5000/50.000	85/3072	100.000	dcs.single_node
64	58,2	5000/60.000	128/5120	100.000	dcs.single_node

Instâncias de DCS principal/em espera

Para cada instância do Redis de DCS principal/em espera, a memória disponível é menor que a de uma instância do Redis de DCS de nó único porque alguma memória é reservada para persistência de dados, conforme mostrado na **Tabela 4-2**. A memória disponível de uma instância principal/em espera pode ser ajustada para suportar tarefas em segundo plano, como persistência de dados e sincronização principal/em espera.

Tabela 4-2 Especificações das instâncias principal/em espera do DCS Redis 3.0

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
2	1,5	5000/50.000	42/512	50.000	dcs.principal_em espera
4	3.2	5000/50.000	64/1536	100.000	dcs.principal_em espera
8	6.4	5000/50.000	64/1536	100.000	dcs.principal_em espera
16	12,8	5000/50.000	85/3072	100.000	dcs.principal_em espera
32	25,6	5000/50.000	85/3072	100.000	dcs.principal_em espera

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
64	51,2	5000/60.000	128/5120	100.000	dcs.principal_em espera

Instâncias de Cluster de Proxy

Além da memória maior, as instâncias de cluster apresentam mais conexões permitidas, maior largura de banda permitida e mais QPS do que as instâncias de nó único e principal/em espera.

Tabela 4-3 Especificações das instâncias do cluster de proxy DCS Redis 3.0

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
64	64	90.000/90.000	600/5120	500.000	dcs.cluster
128	128	180.000/180.000	600/5120	500.000	dcs.cluster
256	256	240.000/240.000	600/5120	500.000	dcs.cluster
512	512	480.000/480.000	600/5120	500.000	dcs.cluster
1024	1024	960.000/960.000	600/5120	500.000	dcs.cluster

NOTA

- Cluster de proxy pago por uso As instâncias do DCS Redis estão disponíveis em 64 GB, 128 GB e 256 GB.
- Cluster de proxy anual/mensal As instâncias do DCS Redis estão disponíveis em 64 GB, 128 GB, 256 GB, 512 GB e 1024 GB.

  Atualmente, apenas a região CN-Hong Kong suporta faturamento anual/mensal. Se você precisar usar esse modo de cobrança em outras regiões, envie um tíquete de serviço no console para solicitar que a equipe técnica ative a função para você em segundo plano.

4.2 Especificações de instância do Redis 4.0 e 5.0

Esta seção descreve as especificações das instâncias do DCS Redis 4.0 e 5.0, incluindo a memória total, a memória disponível, o número máximo de conexões permitidas, a largura de banda máxima/garantida e o desempenho de referência.

As métricas a seguir estão relacionadas às especificações da instância:

- Memória usada: Você pode verificar o uso de memória de uma instância visualizando as métricas **Memory Usage** e **Used Memory**.
- Máximo de conexões: O número máximo de conexões permitidas é o número máximo de clientes que podem ser conectados a uma instância. Para verificar o número de conexões com uma instância, exiba a métrica **Used Memory**.
- O QPS representa consultas por segundo, que é o número de comandos processados por segundo.
- Largura de banda Você pode ver a métrica **Flow Control Times** para verificar se a largura de banda excedeu o limite. Você também pode verificar a métrica **Flow Control Times**. Essa métrica é apenas para referência, pois pode ser maior que 100%. Para obter detalhes, consulte **Por que o uso da largura de banda excede 100%?**

NOTA

- As instâncias do DCS Redis 4.0 e 5.0 estão disponíveis nos tipos de divisão de nó único, principal/em espera, cluster de proxy, cluster do Redis e leitura/gravação.
- Ambos os DCS Redis 4.0 e 5.0 suportam arquiteturas x86 e Arm CPU. Para obter detalhes sobre as diferenças, consulte a descrição a seguir.

Instâncias de nó único

As instâncias do DCS Redis 4.0 ou 5.0 de nó único suportam arquiteturas x86 e Arm CPU. A tabela a seguir lista as especificações.

Tabela 4-4 Especificações de instâncias do DCS Redis 4.0 ou 5.0 de nó único

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
0,125	0,125	10.000/10.000	40/40	80.000	x86: redis.single.xu1.tiny.128 Braço: redis.single.au1.tiny.128

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
0,25	0,25	10.000/10.000	80/80	80.000	x86: redis.single.xu1.tiny.256 Braço: redis.single.au1.tiny.256
0,5	0,5	10.000/10.000	80/80	80.000	x86: redis.single.xu1.tiny.512 Braço: redis.single.au1.tiny.512
1	1	10.000/50.000	80/80	80.000	x86: redis.single.xu1.large.1 Braço: redis.single.au1.large.1
2	2	10.000/50.000	128/128	80.000	x86: redis.single.xu1.large.2 Braço: redis.single.au1.large.2
4	4	10.000/50.000	192/192	80.000	x86: redis.single.xu1.large.4 Braço: redis.single.au1.large.4
8	8	10.000/50.000	192/192	100.000	x86: redis.single.xu1.large.8 Braço: redis.single.au1.large.8

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
16	16	10.000/50.000	256/256	100.000	x86: redis.single.xu1.large.16 Braço: redis.single.au1.large.16
24	24	10,000/50,000	256/256	100.000	x86: redis.single.xu1.large.24 Braço: redis.single.au1.large.24
32	32	10.000/50.000	256/256	100.000	x86: redis.single.xu1.large.32 Braço: redis.single.au1.large.32
48	48	10.000/50.000	256/256	100.000	x86: redis.single.xu1.large.48 Braço: redis.single.au1.large.48
64	64	10.000/50.000	384/384	100.000	x86: redis.single.xu1.large.64 Braço: redis.single.au1.large.64

Instâncias de DCS principal/em espera

As instâncias principal/em espera suportam arquiteturas x86 e Arm CPU. Uma instância pode ter de 2 a 5 réplicas. Por exemplo, as especificações de uma instância baseada em ARM podem ser **Arm | master/standby | 2 replicas** ou **Arm | master/standby | 5 replicas**. Por padrão, uma instância principal/em espera tem um nó principal e duas réplicas (incluindo a réplica principal).

Dado o mesmo tamanho de memória, as diferenças entre instâncias principal/em espera baseadas em x86, instâncias principal/em espera baseadas em Arm e instâncias principal/em espera com várias réplicas são as seguintes:

- A memória disponível, o número máximo de conexões, a largura de banda assegurada/máxima e o QPS são os mesmos.
- Nome da especificação: **Tabela 4-5** lista apenas os nomes de especificação de instâncias baseadas em x86 e Arm. Os nomes das especificações refletem o número de réplicas, por exemplo, redis.ha.au1.large.**r2**.8 (principal/em espera + Arm + 2 réplicas + 8 GB) e redis.ha.au1.large.**r3**.8 (principal/em espera + Arm + 2 réplicas + 8 GB) Foram 3 réplicas foram 8 GB).
- Endereços IP Número de endereços IP ocupados = Número de nós principais x Número de réplicas. Por exemplo:

  2 réplicas: Número de endereços IP ocupados = 1 x 2 = 2

  3 réplicas: Número de endereços IP ocupados = 1 x 3 = 3

Tabela 4-5 Especificações das instâncias principal/em espera do DCS Redis 4.0 ou 5.0

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/Limite) (Contagem)	Largura de banda assegurada/máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
0,125	0,125	10.000/10.000	40/40	80.000	x86: redis.ha.xu1.tiny.r2.128 Braço: redis.ha.au1.tiny.128
0,25	0,25	10.000/10.000	80/80	80.000	x86: redis.ha.xu1.tiny.r2.256 Braço: redis.ha.au1.tiny.256
0,5	0,5	10.000/10.000	80/80	80.000	x86: redis.ha.xu1.tiny.r2.512 Braço: redis.ha.au1.tiny.512

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada/ máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
1	1	10.000/50.000	80/80	80.000	x86: redis.ha.xu1.large.r2.1 Braço: redis.ha.au1.large.1
2	2	10.000/50.000	128/128	80.000	x86: redis.ha.xu1.large.r2.2 Braço: redis.ha.au1.large.2
4	4	10.000/50.000	192/192	80.000	x86: redis.ha.xu1.large.r2.4 Braço: redis.ha.au1.large.4
8	8	10.000/50.000	192/192	100.000	x86: redis.ha.xu1.large.r2.8 Braço: redis.ha.au1.large.8
16	16	10.000/50.000	256/256	100.000	x86: redis.ha.xu1.large.r2.16 Braço: redis.ha.au1.large.16
24	24	10.000/50.000	256/256	100.000	x86: redis.ha.xu1.large.r2.24 Braço: redis.ha.au1.large.24
32	32	10.000/50.000	256/256	100.000	x86: redis.ha.xu1.large.r2.32 Braço: redis.ha.au1.large.32

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/Limite) (Contagem)	Largura de banda assegurada/máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
48	48	10.000/50.000	256/256	100.000	x86: redis.ha.xu1.large.r2.48 Braço: redis.ha.au1.large.48
64	64	10.000/50.000	384/384	100.000	x86: redis.ha.xu1.large.r2.64 Braço: redis.ha.au1.large.64

Instâncias de Cluster de Proxy

Instâncias de cluster proxy suportam arquiteturas de CPU x86 e Arm. **Tabela 4-6** lista as especificações.

As instâncias de cluster de proxy não suportam a personalização de fragmentos e réplicas. Para obter detalhes sobre o número padrão de estilhaços, consulte **Tabela 3-2**. Por padrão, cada estilhaço tem duas réplicas.

Tabela 4-6 Especificações das instâncias de cluster de proxy DCS Redis 4.0 e 5.0

Especificação (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/Limite) (Contagem)	Largura de banda assegurada/máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
4	4	20.000/20.000	1.000/1.000	240.000	x86: redis.proxy.xu1.large.4 Braço: redis.proxy.au1.large.4

Especificação (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/Limite) (Contagem)	Largura de banda assegurada/máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
8	8	30.000/30.000	2.000/2.000	240.000	x86: redis.proxy.xu1.large.8 Braço: redis.proxy.au1.large.8
16	16	30.000/30.000	3.072/3.072	240.000	x86: redis.proxy.xu1.large.16 Braço: redis.proxy.au1.large.16
24	24	30.000/30.000	3.072/3.072	240.000	x86: redis.proxy.xu1.large.24 Braço: redis.proxy.au1.large.24
32	32	30.000/30.000	3.072/3.072	240.000	x86: redis.proxy.xu1.large.32 Braço: redis.proxy.au1.large.32
48	48	60.000/60.000	4608/4608	480.000	x86: redis.proxy.xu1.large.48 Braço: redis.proxy.au1.large.48
64	64	80.000/80.000	6144/6144	640.000	x86: redis.proxy.xu1.large.64 Braço: redis.proxy.au1.large.64

Especificação (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/Limite) (Contagem)	Largura de banda assegurada/máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
96	96	120.000/120.000	9216/9216	960.000	x86: redis.proxy.xu1.large.96 Braço: redis.proxy.au1.large.96
128	128	160.000/160.000	10.000/10.000	1.280.000	x86: redis.proxy.xu1.large.128 Braço: redis.proxy.au1.large.128
192	192	240.000/240.000	10.000/10.000	1.920.000	x86: redis.proxy.xu1.large.192 Braço: redis.proxy.au1.large.192
256	256	320.000/320.000	10.000/10.000	> 2.000.000	x86: redis.proxy.xu1.large.256 Braço: redis.proxy.au1.large.256
384	384	480.000/480.000	10.000/10.000	> 2.000.000	x86: redis.proxy.xu1.large.384 Braço: redis.proxy.au1.large.384
512	512	640.000/640.000	10.000/10.000	> 2.000.000	x86: redis.proxy.xu1.large.512 Braço: redis.proxy.au1.large.512

Especificação (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/Limite) (Contagem)	Largura de banda assegurada/máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
768	768	640.000/640.000	10.000/10.000	> 2.000.000	x86: redis.proxy.xu1.large.768 Braço: redis.proxy.au1.large.768
1024	1024	640.000/640.000	10.000/10.000	> 2.000.000	x86: redis.proxy.xu1.large.1024 Braço: redis.proxy.au1.large.1024

Instâncias do Cluster do Redis

Instâncias de cluster proxy suportam arquiteturas de CPU x86 e Arm. Uma instância pode ter de 2 a 5 réplicas. Por exemplo, as especificações da instância podem ser **Redis Cluster | 1 replica** ou **Redis Cluster | 5 replicas**. Por padrão, uma instância do Cluster do Redis tem duas réplicas. Uma instância do Cluster do Redis com apenas 1 réplica indica que a quantidade de réplicas foi diminuída.

Dado o mesmo tamanho de memória, as diferenças entre as instâncias do Redis Cluster baseadas em x86, as instâncias do Redis Cluster baseadas em ARM e as instâncias do Redis Cluster com várias réplicas são as seguintes:

- A memória disponível, quantidade de estilhaços (quantidade de nó principal), número máximo de conexões, largura de banda assegurada/máxima e QPS são os mesmos.

  NOTA

  A largura de banda máxima e a largura de banda assegurada de um cluster do Redis é para toda a instância, e não para um único fragmento.

- Nome da especificação: **Tabela 4-7** lista apenas os nomes de especificação de instâncias baseadas em x86 e ARM com 2 réplicas. Os nomes de especificação refletem o número de réplicas, por exemplo, redis.cluster.au1.large.**r2**.24 . (Redis Cluster + Braço + 2 réplicas + 24 GB) e redis.cluster.au1.large.**r3**.24 (Redis Cluster + Braço + 3 réplicas + 24 GB).
- Número de endereços IP ocupados = Número de nós principais x Número de réplicas. Por exemplo:

  24 GB + Redis Cluster + 3 réplicas: Número de endereços IP ocupados = 3 x 3 = 9
- Memória disponível por nó = Memória disponível de instância/Quantidade de nó principal

Por exemplo, uma instância de 24 GB baseada em x86 tem 24 GB de memória disponível e 3 nós principais. A memória disponível por nó é 24/3 = 8 GB.

- Limite máximo de conexões por nó = Limite máximo de conexões/Quantidade do nó principal Por exemplo:

  Por exemplo, uma instância de 24 GB baseada em x86 tem 3 nós principais e o limite máximo de conexões é 150 000. O limite máximo de conexões por nó = 150.000/3 = 50 000.

Tabela 4-7 Especificações das instâncias de cluster de proxy DCS Redis 4.0 e 5.0

Memória Total (em GB)	Memória Disponível (em GB)	Estilhaços (principal Nodes)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada/ máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
4	4	3	30.000 / 150.000	2304/2304	240.000	x86: redis.cluster.xu1.large.r2.4 Braço: redis.cluster.au1.large.r2.4
8	8	3	30.000 / 150.000	2304/2304	240.000	x86: redis.cluster.xu1.large.r2.8 Braço: redis.cluster.au1.large.r2.8
16	16	3	30.000 / 150.000	2304/2304	240 000	x86: redis.cluster.xu1.large.r2.16 Braço: redis.cluster.au1.large.r2.16
24	24	3	30.000 / 150.000	2304/2304	300.000	x86: redis.cluster.xu1.large.r2.24 Braço: redis.cluster.au1.large.r2.24

Memória Total (em GB)	Memória Disponível (em GB)	Estilhaços (principal Nodes)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada/ máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
32	32	3	30.000 / 150.000	2304/2304	300.000	x86: redis.cluster.xu1.large.r2.32 Braço: redis.cluster.au1.large.r2.32
48	48	6	60.000 / 300.000	4608/4608	> 300.000	x86: redis.cluster.xu1.large.r2.48 Braço: redis.cluster.au1.large.r2.48
64	64	8	80.000 / 400.000	6144/6144	500.000	x86: redis.cluster.xu1.large.r2.64 Braço: redis.cluster.au1.large.r2.64
96	96	12	120.000 / 600.000	9216/9216	> 500.000	x86: redis.cluster.xu1.large.r2.96 Braço: redis.cluster.au1.large.r2.96
128	128	16	160.000 / 800.000	12.288/12.288	1.000.000	x86: redis.cluster.xu1.large.r2.128 Braço: redis.cluster.au1.large.r2.128
192	192	24	240 000 / 1.200.000	18.432/18.432	> 1.000.000	x86: redis.cluster.xu1.large.r2.192 Braço: redis.cluster.au1.large.r2.192

Memória Total (em GB)	Memória Disponível (em GB)	Estilhaços (principal Nodes)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada/ máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
256	256	32	320.000 / 1.600.000	24.576/24.576	> 2.000.000	x86: redis.cluster.xu1.large.r2.256 Braço: redis.cluster.au1.large.r2.256
384	384	48	480.000 / 2.400.000	36.864/36.864	> 2.000.000	x86: redis.cluster.xu1.large.r2.384 Braço: redis.cluster.au1.large.r2.384
512	512	64	640.000 / 3.200.000	49.152/49.152	> 2.000.000	x86: redis.cluster.xu1.large.r2.512 Braço: redis.cluster.au1.large.r2.512
768	768	96	960 000 / 4.800.000	73.728/73.728	> 2.000.000	x86: redis.cluster.xu1.large.r2.768 Braço: redis.cluster.au1.large.r2.768
1024	1024	128	1.280.000 / 6.400.000	98.304/98.304	> 2.000.000	x86: redis.cluster.xu1.large.r2.1024 Braço: redis.cluster.au1.large.r2.1024

Instâncias de divisão de leitura/gravação

- Atualmente, as instâncias de divisão de leitura/gravação suportam apenas a arquitetura da CPU x86. **Tabela 4-8** lista as especificações.
- O número máximo de conexões de uma instância de divisão de leitura/gravação do DCS Redis 4.0 ou 5.0 não pode ser modificado.

- Limite de largura de banda por servidor Redis (MB/s) = Limite total de largura de banda (MB/s)/Número de réplicas (incluindo principais)
- Desempenho de referência por nó (QPS) = Desempenho de referência (QPS)/Número de réplicas (incluindo principais)
- Ao usar instâncias de divisão de leitura/gravação, observe o seguinte:

  a. As solicitações de leitura são enviadas às réplicas. Há um atraso quando os dados são sincronizados do principal para as réplicas.

     Se seus serviços forem sensíveis ao atraso, não use instâncias de divisão de leitura/gravação. Em vez disso, use instâncias principal/em espera ou cluster.

  b. A divisão de leitura/gravação é adequada quando há mais solicitações de leitura do que solicitações de gravação. Se houver muitas solicitações de gravação, o principal e as réplicas podem ser desconectadas ou a sincronização de dados entre elas pode falhar após a desconexão. Como resultado, o desempenho de leitura se deteriora.

     Se os seus serviços forem pesados em gravação, use instâncias principal/em espera ou de cluster.

  c. Se uma réplica estiver com defeito, leva algum tempo para sincronizar todos os dados do principal. Durante a sincronização, a réplica não fornece serviços e o desempenho de leitura da instância se deteriora.

     Para reduzir o impacto da falha, use uma instância com menos de 32 GB de memória. Quanto menor a memória, menor o tempo para sincronização completa de dados entre o principal e as réplicas, e menor o impacto da interrupção.

Tabela 4-8 Especificações de instâncias de divisão de leitura/gravação do DCS Redis 4.0 ou 5.0

Especificação	Memória Disponível (GB)	Réplicas (incluindo Principais)	Máx. Conexões (Padrão/ Limite)	Limite de largura de banda (MB/s)	Limite de largura de banda por servidor Redis (MB/s)	Desempenho de Referência (QPS)	Desempenho de referência por nó (QPS)	Código de especificação (spec_code na API)
8	8	2	20.000	192	96	160.000	80.000	redis.ha.xu1.large.p2.8
8	8	3	30.000	288	96	240.000	80.000	redis.ha.xu1.large.p3.8
8	8	4	40.000	384	96	320.000	80.000	redis.ha.xu1.large.p4.8
8	8	5	50.000	480	96	400.000	80.000	redis.ha.xu1.large.p5.8

Especificação	Memória Disponível (GB)	Réplicas (incluindo Principais)	Máx. Conexões (Padrão/ Limite)	Limite de largura de banda (MB/s)	Limite de largura de banda por servidor Redis (MB/s)	Desempenho de Referência (QPS)	Desempenho de referência por nó (QPS)	Código de especificação (spec_code na API)
8	8	6	60.000	576	96	480.000	80.000	redis.ha.xu1.large.p6.8
16	16	2	20.000	192	96	160.000	80.000	redis.ha.xu1.large.p2.16
16	16	3	30.000	288	96	240.000	80.000	redis.ha.xu1.large.p3.16
16	16	4	40.000	384	96	320.000	80.000	redis.ha.xu1.large.p4.16
16	16	5	50.000	480	96	400.000	80.000	redis.ha.xu1.large.p5.16
16	16	6	60.000	576	96	480.000	80.000	redis.ha.xu1.large.p6.16
32	32	2	20.000	192	96	160.000	80.000	redis.ha.xu1.large.p2.32
32	32	3	30.000	288	96	240.000	80.000	redis.ha.xu1.large.p3.32
32	32	4	40.000	384	96	320.000	80.000	redis.ha.xu1.large.p4.32

Especificação	Memória Disponível (GB)	Réplicas (incluindo Principais)	Máx. Conexões (Padrão/ Limite)	Limite de largura de banda (MB/s)	Limite de largura de banda por servidor Redis (MB/s)	Desempenho de Referência (QPS)	Desempenho de referência por nó (QPS)	Código de especificação (spec_code na API)
32	32	5	50.000	480	96	400.000	80.000	redis.ha.xu1.large.p5.32
32	32	6	60.000	576	96	480.000	80.000	redis.ha.xu1.large.p6.32

4.3 Especificações de instância (OBT) do Redis 6.0

Esta seção descreve as especificações da instância do DCS Redis 6.0, incluindo a memória total, a memória disponível, o número máximo de conexões permitidas, a largura de banda máxima/garantida e o desempenho de referência.

As métricas a seguir estão relacionadas às especificações da instância:

- Memória usada: Você pode verificar o uso de memória de uma instância exibindo as métricas **Memory Usage** e **Used Memory**.
- Máximo de conexões: O número máximo de conexões permitidas é o número máximo de clientes que podem ser conectados a uma instância. Para verificar o número de conexões com uma instância, exiba a métrica **Connected Clients**.
- O QPS representa consultas por segundo, que é o número de comandos processados por segundo. Para obter detalhes sobre o teste QPS, consulte o **Papel Branco de Desempenho**.

O DCS for Redis 6.0 é fornecido nas edições básica, corporativa (desempenho) e corporativa (armazenamento). Eles estão disponíveis para OBT na região CN North-Beijing4.

Edição Básica

Atualmente, a edição básica do DCS for Redis 6.0 é compatível com instâncias principal/em espera e de nó único baseadas em CPUs x86.

Tabela 4-9 Especificações das instâncias principal/em espera do DCS Redis 6.0 Basic Edition

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
4	4	10.000/50.000	192/192	280.000	redis.ha.xu1.large.r2.v6.4
8	8	10.000/50.000	192/192	300.000	redis.ha.xu1.large.r2.v6.8
16	16	10.000/50.000	256/256	300.000	redis.ha.xu1.large.r2.v6.16

Tabela 4-10 Especificações de instâncias do DCS Redis 6.0 Basic Edition de nó único

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
4	4	10.000/50.000	192/192	280.000	redis.single.xu1.large.v6.4
8	8	10.000/50.000	192/192	300.000	redis.single.xu1.large.v6.8
16	16	10.000/50.000	256/256	300.000	redis.single.xu1.large.v6.16

Edição Profissional (Performance)

Atualmente, a edição profissional (desempenho) do DCS for Redis 6.0 oferece suporte a instâncias principal/em espera baseadas em CPUs x86.

Tabela 4-11 Especificações das instâncias do DCS Redis 6.0 professional (desempenho) edition

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
4	4	10.000/50.000	768/768	200.000	redis.ha.xu1.large.enthp.4

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
8	8	10.000/50.000	1536/1536	400.000	redis.ha.xu1.large.enthp.8
16	16	10.000/50.000	1536/1536	400.000	redis.ha.xu1.large.enthp.16
32	32	10.000/50.000	1536/1536	400.000	redis.ha.xu1.large.enthp.32
64	64	10.000/50.000	1536/1536	400.000	redis.ha.xu1.large.enthp.64

Edição Profissional (Armazenamento)

Atualmente, a edição profissional (armazenamento) do DCS for Redis 6.0 suporta instâncias principal/em espera baseadas em CPUs x86.

Tabela 4-12 Especificações das instâncias do DCS Redis 6.0 professional (storage) edition

Memória Total (em GB)	Armazenamento Máximo (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada/ máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
8	64	10.000/50.000	768/768	70.000	redis.ha.xu1.large.entst.8
16	128	10.000/50.000	768/768	70.000	redis.ha.xu1.large.entst.16
32	256	10.000/50.000	768/768	70.000	redis.ha.xu1.large.entst.32

4.4 Especificações de instância do Memcached (indisponível em breve)

NOTA

O DCS for Memcached está prestes a ficar indisponível e não é mais vendido em algumas regiões. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.

Esta seção descreve as especificações da instância do Memcached DCS, incluindo a memória total, a memória disponível, o número máximo de conexões permitidas, a largura de banda máxima/garantida e o desempenho de referência.

O número máximo de conexões permitidas é o número máximo de clientes que podem ser conectados a uma instância. Para verificar o número de conexões com uma instância, exiba a métrica **Connected Clients**.

O QPS representa consultas por segundo, que é o número de comandos processados por segundo.

NOTA

As instâncias do Memcached DCS estão disponíveis nos tipos de nó único e principal/em espera.

Instâncias de nó único

Para cada instância do Memcached DCS de nó único, a memória disponível é menor do que a memória total, pois alguma memória é reservada para sobrecargas do sistema, conforme mostrado na **Tabela 4-13**.

Tabela 4-13 Especificações de instâncias de Memcached DCS de nó único

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
2	1,5	5.000/50.000	42/128	50.000	dcs.memcached.single_node
4	3.2	5.000/50.000	64/192	100.000	dcs.memcached.single_node
8	6,8	5.000/50.000	64/192	100.000	dcs.memcached.single_node
16	13,6	5.000/50.000	85/256	100.000	dcs.memcached.single_node
32	27,2	5.000/50.000	85/256	100.000	dcs.memcached.single_node
64	58,2	5.000/50.000	128/384	100.000	dcs.memcached.single_node

Instâncias de DCS principal/em espera

Para cada instância de Memcached DCS principal/em espera, a memória disponível é menor do que a memória total porque alguma memória é reservada para persistência de dados, conforme mostrado na . A memória disponível de uma instância principal/em espera pode ser ajustada para suportar tarefas em segundo plano, como persistência de dados e sincronização principal/em espera.

Tabela 4-14 Especificações das instâncias principal/em espera do DCS Redis 3.0

Memória Total (em GB)	Memória Disponível (em GB)	Máx. Conexões (Padrão/ Limite) (Contagem)	Largura de banda assegurada /máxima (Mbit/s)	Desempenho de referência (QPS)	Código de especificação (spec_code na API)
2	1,5	5.000/50.000	42/128	50.000	dcs.memcached.master_em espera
4	3.2	5.000/50.000	64/192	100.000	dcs.memcached.master_em espera
8	6,8	5.000/50.000	64/192	100.000	dcs.memcached.master_em espera
16	13,6	5.000/50.000	85/256	100.000	dcs.memcached.master_em espera
32	27,2	5.000/50.000	85/256	100.000	dcs.memcached.master_em espera
64	58,2	5.000/50.000	128/384	100.000	dcs.memcached.master_em espera

5 Compatibilidade de comandos

5.1 Comandos do Redis 3.0

O DCS for Redis 3.0 foi desenvolvido com base no Redis 3.0.7 e é compatível com protocolos e comandos de código aberto. Esta seção descreve a compatibilidade do DCS for Redis 3.0 com comandos do Redis, incluindo comandos suportados, comandos desabilitados, scripts e comandos não suportados de versões posteriores do Redis e restrições ao uso de comandos.

NOTA

O DCS for Redis 3.0 não é mais fornecido. Em vez disso, você pode usar o DCS para Redis 4.0 ou 5.0.

As instâncias do DCS Redis são compatíveis com a maioria dos comandos do Redis. Qualquer cliente compatível com o protocolo Redis pode acessar o DCS.

- Por motivos de segurança, alguns comandos do Redis são desativados no DCS, conforme listado em **Comandos desabilitados pelo DCS para o Redis 3.0**.
- Alguns comandos do Redis são suportados por instâncias de DCS de cluster para operações de várias chaves no mesmo slot. Para mais detalhes, consulte **Restrições de Comando**.
- Alguns comandos do Redis têm restrições de uso, que são descritas em **Outras restrições de uso de comandos**.

Comandos suportados pelo DCS for Redis 3.0

A lista a seguir apresenta os comandos suportados pelo DCS para o Redis 3.0. Para obter detalhes sobre a sintaxe do comando, visite o **site oficial do Redis**. Por exemplo, para exibir detalhes sobre o comando **SCAN**, digite **SCAN** na caixa de pesquisa **desta página**.

NOTA

- Os comandos disponíveis desde versões posteriores do Redis não são suportados por instâncias de versões anteriores. Execute um comando no redis-cli para verificar se ele é suportado pelo DCS for Redis. Se a mensagem "(error) ERR comando desconhecido" for retornada, o comando não é suportado.
- Os seguintes comandos listados nas tabelas não são suportados por instâncias de cluster de proxy:
  - **List** grupo: **BLPOP**, **BRPOP** e **BRPOPLRUSH**
  - Comandos **CLIENT** no grupo **Server**: **CLIENT KILL**, **CLIENT GETNAME**, **CLIENT LIST**, **CLIENT SETNAME**, **CLIENT PAUSE** e **CLIENT REPLY**.
  - Grupo de **Server**: **MONITOR**
  - Grupo de **Transactions**: **UNWATCH** e **WATCH**
  - Grupo **Key**: **RANDOMKEY** (para instâncias antigas)

Tabela 5-1 Comandos suportados pelas instâncias do DCS Redis 3.0 (1)

Chaves	String	Hash	Lista	Set	Conjunto Ordenado	Servidor
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENT KILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT LIST
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT GETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENT SETNAME
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIG GET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR

Chaves	String	Hash	Lista	Set	Conjunto Ordenado	Servidor
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	-	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	-	RPOPLPUSH	SSCAN	ZINTERSTORE	-
OBJECT	PSETEX	-	RPUSH	-	ZSCAN	-
-	SET	-	RPUSHX	-	ZRANGEBYLEX	-
-	SETBIT	-	-	-	-	-
-	SETEX	-	-	-	-	-
-	SETNX	-	-	-	-	-
-	SETRANGE	-	-	-	-	-
-	STRLEN	-	-	-	-	-

Tabela 5-2 Comandos suportados pelas instâncias do DCS Redis 3.0 (2)

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER

Comandos desabilitados pelo DCS para o Redis 3.0

A lista a seguir apresenta os comandos suportados pelo DCS para o Redis 3.0.

Tabela 5-3 Comandos do Redis desabilitados em instâncias do Redis 3.0 de nó único e de DCS principal/em espera

Chaves	Servidor
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	Comandos DEBUG
-	COMMAND
-	SAVE
-	BGSAVE
-	BGREWRITEAOF

Tabela 5-4 Comandos do Redis desativados em instâncias do Redis 3.0 do Cluster de Proxy DCS

Chaves	Servidor	Lista	Transações	Ligação	Aglomerado	codis
MIGRATE	SLAVEOF	BLPOP	DISCARD	SELECT	CLUSTER	TIME
MOVE	SHUTDOWN	BRPOP	EXEC	-	-	SLOTSINFO
-	LASTSAVE	BRPOPLPUSH	MULTI	-	-	SLOTSDEL
-	Comandos DEBUG	-	UNWATCH	-	-	SLOTSMGRTSLOT
-	COMMAND	-	WATCH	-	-	SLOTSMGRTONE
-	SAVE	-	-	-	-	SLOTSCHECK
-	BGSAVE	-	-	-	-	SLOTSMGRTTAGSLOT
-	BGREWRITEAOF	-	-	-	-	SLOTSMGRTTAGONE
-	SYNC	-	-	-	-	-
-	PSYNC	-	-	-	-	-

Chaves	Servidor	Lista	Transações	Ligação	Aglomerado	codis
-	MONITOR	-	-	-	-	-
-	Comandos CLIENT	-	-	-	-	-
-	OBJECT	-	-	-	-	-
-	ROLE	-	-	-	-	-

5.2 Comandos do Redis 4.0

O DCS for Redis 4.0 foi desenvolvido com base no Redis 4.0.14 e é compatível com protocolos e comandos de código aberto. Esta seção descreve a compatibilidade do DCS para Redis 4.0 com comandos KeyDB, incluindo comandos suportados e desabilitados.

As instâncias do DCS Redis são compatíveis com a maioria dos comandos do Redis. Qualquer cliente compatível com o protocolo Redis pode acessar o DCS.

- Por motivos de segurança, alguns comandos do Redis são desativados no DCS, conforme listado em **Comandos desabilitados pelo DCS for o Redis 4.0**.
- Alguns comandos do Redis são suportados por instâncias de DCS de cluster para operações de várias chaves no mesmo slot. Para mais detalhes, consulte **Restrições de Comando**.
- Alguns comandos do Redis têm restrições de uso, que são descritas em **Outras restrições de uso de comandos**.

Comandos suportados pelo DCS for Redis 4.0

- **Tabela 5-5** e **Tabela 5-6** liste os comandos do Redis suportados pelas instâncias do DCS Redis 4.0.
- **Tabela 5-7** e **Tabela 5-8** liste os comandos do Redis suportados pelas instâncias do Cluster de Proxy DCS Redis 4.0.
- **Tabela 5-9** e **Tabela 5-10** liste os comandos do Redis suportados por instâncias de divisão de leitura/gravação do DCS Redis 4.0.

Para obter detalhes sobre a sintaxe do comando, visite o **site oficial do Redis**. Por exemplo, para exibir detalhes sobre o comando **SCAN**, digite **SCAN** na caixa de pesquisa **desta página**.

NOTA

- Os comandos disponíveis desde versões posteriores do Redis não são suportados por instâncias de versões anteriores. Execute um comando no redis-cli para verificar se ele é suportado pelo DCS for Redis. Se a mensagem "(error) ERR comando desconhecido" for retornada, o comando não é suportado.
- Para instâncias do DCS Redis 4.0 no modo Cluster do Redis, certifique-se de que todos os comandos em um pipeline sejam executados no mesmo fragmento.

Tabela 5-5 Comandos suportados pelas instâncias do DCS Redis 4.0 (1)

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENT KILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT LIST
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT GETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENT SETNAME
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIG GET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	HSTRLEN	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	SWAPDB
OBJECT	PSETEX	-	RPUSH	SPOP	ZSCAN	MEMORY
PEXPIRE	SET	-	RPUSHX	-	ZRANGEBYLEX	CONFIG
PEXPIREAT	SETBIT	-	LPUSH	-	ZLEXCOUNT	-

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
-	SETEX	-	-	-	ZREMRANGEBYSCORE	-
-	SETNX	-	-	-	ZREM	-
-	SETRANGE	-	-	-	-	-
-	STRLEN	-	-	-	-	-
-	BITFIELD	-	-	-	-	-

Tabela 5-6 Comandos suportados pelas instâncias do DCS Redis 4.0 (2)

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER

Tabela 5-7 Comandos suportados pelas instâncias do Cluster de Proxy DCS Redis 4.0 (1)

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	CONTADE BIT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTE	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	ROLE
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	MEMORY
RENAME	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	COMMAND
RENAMENX	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	COMMAND COUNT
RESTORE	INCR	HSET	LREM	SPOP	ZREVRANGE	COMMAND GETKEYS
SORT	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	COMMAND INFO
TTL	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	CONFIG GET
TYPE	MGET	HSCAN	RPOP	SUNION	ZSCORE	CONFIG RESETSTAT
SCAN	MSET	HSTRLEN	RPOPLPUSH	SUNIONSTORE	ZUNIONSTORE	CONFIG REWRITE
OBJECT	MSETNX	HLEN	RPUSH	SSCAN	ZINTERSTORE	CONFIG SET
PEXPIRE	PSETEX	HKEYS	RPUSHX	SPOP	ZSCAN	-
PEXPIREAT	SET	-	LPUSH	-	ZRANGEBYLEX	-
EXPIREAT	SETBIT	-	-	-	ZLEXCOUNT	-

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
KEYS	SETEX	-	-	-	ZREMRANGEBYSCORE	-
TOUCH	SETNX	-	-	-	ZREM	-
UNLINK	SETRANGE	-	-	-	ZREMRANGEBYLEX	-
-	STRLEN	-	-	-	ZREVRANGEBYLEX	-
-	BITFIELD	-	-	-	-	-
-	GETBIT	-	-	-	-	-

Tabela 5-8 Comandos suportados pelas instâncias do Cluster de Proxy DCS Redis 4.0 (2)

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSBYMEMBER
-	-	-	CLIENT LIST	SCRIPT DEBUG YES|SYNC|NO	GEOSEARCH
-	-	-	CLIENTE GETNAME	-	GEOSEARCHSTORE

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos
-	-	-	CLIENTE SETNAME	-	-

Tabela 5-9 Comandos suportados por instâncias de divisão de leitura/gravação do DCS Redis 4.0 (1)

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCONT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	MONITOR
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	SLOWLOG
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	ROLE
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	SWAPDB
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	MEMORY
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	COMMAND
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	COMMAND COUNT

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	COMMAND GETKEYS
TYPE	MSET	HSTRLEN	RPOPLPUSH	SUNIONSTORE	ZUNIONSTORE	COMMAND INFO
SCAN	MSETNX	HLEN	RPUSH	SSCAN	ZINTERSTORE	CONFIG GET
OBJECT	PSETEX	-	RPUSHX	SPOP	ZSCAN	CONFIG RESETSTAT
PEXPIRE	SET	-	LPUSH	-	ZRANGEBYLEX	CONFIG REWRITE
PEXPIREAT	SETBIT	-	-	-	ZLEXCOUNT	CONFIG SET
EXPIREAT	SETEX	-	-	-	ZREMRANGEBYSCORE	-
KEYS	SETNX	-	-	-	ZREM	-
TOUCH	SETRANGE	-	-	-	ZREMRANGEBYLEX	-
UNLINK	STRLEN	-	-	-	ZREVRANGEBYLEX	-
-	BITFIELD	-	-	-	-	-
-	GETBIT	-	-	-	-	-

Tabela 5-10 Comandos suportados por instâncias de divisão de leitura/gravação do DCS Redis 4.0 (2)

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSBYMEMBER
-	-	-	CLIENT LIST	SCRIPT DEBUG YES|SYNC|NO	GEOSEARCH
-	-	-	CLIENT GETNAME	-	GEOSEARCHSTORE
-	-	-	CLIENT SETNAME	-	-

Comandos desabilitados pelo DCS for o Redis 4.0

A lista a seguir apresenta os comandos suportados pelo DCS for o Redis 4.0.

Tabela 5-11 Comandos do Redis desabilitados em instâncias do Redis 4.0 de nó único e de DCS principal/em espera

Chaves	Servidor
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	Comandos DEBUG
-	COMMAND
-	SAVE
-	BGSAVE
-	BGREWRITEAOF

Chaves	Servidor
-	SINCRONIZAÇÃO
-	PSYNC

Tabela 5-12 Comandos do Redis desativados em instâncias do Redis 4.0 do Cluster de Proxy DCS

Chaves	Servidor	Conjunto Ordenado	Aglomerado
MIGRATE	BGREWRITEAOF	BZPOPMAX	READONLY
MOVE	BGSAVE	BZPOPMIN	READWRITE
RANDOMKEY	Comandos CLIENT	ZPOPMAX	-
WAIT	DEBUG OBJECT	ZPOPMIN	-
-	DEBUG SEGFAULT	-	-
-	LASTSAVE	-	-
-	PSYNC	-	-
-	SAVE	-	-
-	SHUTDOWN	-	-
-	SLAVEOF	-	-
-	Comandos LATENCY	-	-
-	Comandos MODULE	-	-
-	LOLWUT	-	-
-	SWAPDB	-	-
-	REPLICAOF	-	-
-	SYNC	-	-

Tabela 5-13 Comandos do Redis desativados em instâncias do Redis 4.0 do Cluster de Proxy DCS

Chaves	Servidor	Aglomerado
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS
-	LASTSAVE	CLUSTER ADDSLOTS

Chaves	Servidor	Aglomerado
-	Comandos DEBUG	CLUSTER DELSLOTS
-	COMMAND	CLUSTER SETSLOT
-	SAVE	CLUSTER BUMPEPOCH
-	BGSAVE	CLUSTER SAVECONFIG
-	BGREWRITEAOF	CLUSTER FORGET
-	SYNC	CLUSTER REPLICATE
-	PSYNC	CLUSTER COUNT-FAILURE-REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

Tabela 5-14 Comandos do Redis desativados em instâncias de divisão de leitura/gravação do DCS Redis 4.0

Aglomerado	Chaves	Servidor	Conjunto Ordenado
READONLY	MIGRATE	BGREWRITEAOF	BZPOPMAX
READWRITE	WAIT	BGSAVE	BZPOPMIN
-	-	DEBUG OBJECT	ZPOPMAX
-	-	DEBUG SEGFAULT	ZPOPMIN
-	-	LASTSAVE	-
-	-	LOLWUT	-
-	-	MODULE LIST/ LOAD/UNLOAD	-
-	-	PSYNC	-
-	-	REPLICAOF	-
-	-	SAVE	-
-	-	SHUTDOWN [NOSAVE|SAVE]	-
-	-	SLAVEOF	-
-	-	SWAPDB	-

Aglomerado	Chaves	Servidor	Conjunto Ordenado
-	-	SYNC	-

5.3 Comandos do Redis 5.0

O DCS for Redis 5.0 foi desenvolvido com base no Redis 5.0.9 e é compatível com protocolos e comandos de código aberto. Esta seção descreve a compatibilidade do DCS for Redis 5.0 com comandos KeyDB, incluindo comandos suportados e desabilitados.

As instâncias do DCS Redis são compatíveis com a maioria dos comandos do Redis. Qualquer cliente compatível com o protocolo Redis pode acessar o DCS.

- Por motivos de segurança, alguns comandos do Redis são desativados no DCS, conforme listado em **Comandos desabilitados pelo DCS for o Redis 5.0**.
- Alguns comandos do Redis são suportados por instâncias de DCS de cluster para operações de várias chaves no mesmo slot. Para mais detalhes, consulte **Restrições de Comando**.
- Alguns comandos do Redis têm restrições de uso, que são descritas em **Outras restrições de uso de comandos**.

Comandos suportados pelo DCS for Redis 5.0

- **Tabela 5-15** e **Tabela 5-16** liste os comandos suportados pelo DCS para o Redis 5.0.
- **Tabela 5-17** e **Tabela 5-18** liste os comandos suportados pelas instâncias do Proxy Cluster DCS for Redis 5.0.
- **Tabela 5-19** e **Tabela 5-20** liste os comandos do Redis suportados pelas instâncias de divisão de leitura/gravação do DCS Redis 5.0.

Para obter detalhes sobre a sintaxe do comando, visite o **site oficial do Redis**. Por exemplo, para exibir detalhes sobre o comando **SCAN**, digite **SCAN** na caixa de pesquisa **desta página**.

NOTA

- Os comandos disponíveis desde versões posteriores do Redis não são suportados por instâncias de versões anteriores. Execute um comando no redis-cli para verificar se ele é suportado pelo DCS for Redis. Se a mensagem "(error) ERR comando desconhecido" for retornada, o comando não é suportado.
- Para instâncias do DCS Redis 5.0 no modo Cluster do Redis, certifique-se de que todos os comandos em um pipeline sejam executados no mesmo fragmento.

Tabela 5-15 Comandos suportados pelas instâncias do DCS Redis 5.0 (1)

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENTKILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENTLIST
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENTGETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENTESETNAME
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIGGET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	HSTRLEN	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	SWAPDB
OBJECT	PSETEX	-	RPUSH	SPOP	ZSCAN	MEMORY
PEXPIREAT	SET	-	RPUSHX	-	ZRANGEBYLEX	CONFIG
PEXPIRE	SETBIT	-	LPUSH	-	ZLEXCOUNT	-
-	SETEX	-	-	-	ZPOPMIN	-
-	SETNX	-	-	-	ZPOPMAX	-

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
-	SETRANGE	-	-	-	ZREMRANGEBYSCORE	-
-	STRLEN	-	-	-	ZREM	-
-	BITFIELD	-	-	-	-	-

Tabela 5-16 Comandos suportados pelas instâncias do DCS Redis 5.0 (2)

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos	Córrego
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD	XACK
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH	XADD
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XCLAIM
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST	XDEL
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS	XGROUP
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER	XINFO
-	-	-	-	-	-	XLEN
-	-	-	-	-	-	XPENDING
-	-	-	-	-	-	XRANGE
-	-	-	-	-	-	XREAD
-	-	-	-	-	-	XREADGROUP
-	-	-	-	-	-	XREVRANGE
-	-	-	-	-	-	XTRIM

Tabela 5-17 Comandos suportados pelas instâncias do Cluster de Proxy DCS Redis 5.0 (1)

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	ROLE
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	MEMORY
RENAME	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	COMMAND
RENAMENX	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	COMMAND COUNT
RESTORE	INCR	HSET	LREM	SPOP	ZREVRANGE	COMMAND GETKEYS
SORT	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	COMMAND INFO
TTL	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	CONFIG GET
TYPE	MGET	HSCAN	RPOP	SUNION	ZSCORE	CONFIG RESETSTAT
SCAN	MSET	HSTRLEN	RPOPLPUSH	SUNIONSTORE	ZUNIONSTORE	CONFIG REWRITE

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
OBJECT	MSETNX	HLEN	RPUSH	SSCAN	ZINTERSTORE	CONFIG SET
PEXPIRE	PSETEX	HKEYS	RPUSHX	SPOP	ZSCAN	-
PEXPIREAT	SET	-	LPUSH	-	ZRANGEBYLEX	-
EXPIREAT	SETBIT	-	-	-	ZLEXCOUNT	-
KEYS	SETEX	-	-	-	ZREMRANGEBYSCORE	-
MIGRATE	SETNX	-	-	-	ZREM	-
UNLINK	SETRANGE	-	-	-	ZREMRANGEBYLEX	-
TOUCH	STRLEN	-	-	-	ZPOPMAX	-
-	BITFIELD	-	-	-	ZPOPMIN	-
-	GETBIT	-	-	-	BZPOPMAX	-
-	-	-	-	-	BZPOPMIN	-
-	-	-	-	-	ZREVRANGEBYLEX	-

Tabela 5-18 Comandos suportados pelas instâncias do Cluster de Proxy DCS Redis 5.0 (2)

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSBYMEMBER
-	-	-	CLIENT LIST	SCRIPT DEBUG YES|SYNC|NO	GEOSEARCH
-	-	-	CLIENT GETNAME	-	GEOSEARCHSTORE
-	-	-	CLIENTE SETNAME	-	-

Tabela 5-19 Comandos suportados por instâncias de divisão de leitura/gravação do DCS Redis 5.0 (1)

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	MONITOR
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	SLOWLOG

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	ROLE
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	SWAPDB
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	MEMORY
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	COMMAND
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	COMMAND COUNT
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	COMANDO GETKEYS
TYPE	MSET	HSTRLEN	RPOPLPUSH	SUNIONSTORE	ZUNIONSTORE	COMMAND INFO
SCAN	MSETNX	HLEN	RPUSH	SSCAN	ZINTERSTORE	CONFIG GET
OBJECT	PSETEX	-	RPUSHX	SPOP	ZSCAN	CONFIG RESETSTAT
PEXPIRE	SET	-	LPUSH	-	ZRANGEBYLEX	CONFIG REWRITE
PEXPIREAT	SETBIT	-	-	-	ZLEXCOUNT	CONFIG SET
EXPIREAT	SETEX	-	-	-	ZREMRANGEBYSCORE	-
KEYS	SETNX	-	-	-	ZREM	-
MIGRATE	SETRANGE	-	-	-	ZREMRANGEBYLEX	-
UNLINK	STRLEN	-	-	-	BZPOPMAX	-

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
TOUCH	BITFIELD	-	-	-	BZPOPMIN	-
-	GETBIT	-	-	-	ZPOPMAX	-
-	-	-	-	-	ZPOPMIN	-
-	-	-	-	-	ZREVRANGEBYLEX	-

Tabela 5-20 Comandos suportados por instâncias de divisão de leitura/gravação do DCS Redis 5.0 (2)

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	CLIENT KILL	SCRIPT LOAD	GEORADIUSBYMEMBER
-	-	-	CLIENT LIST	SCRIPT DEBUG YES|SYNC|NO	GEOSEARCH
-	-	-	CLIENTE GETNAME	-	GEOSEARCHSTORE
-	-	-	CLIENTE SETNAME	-	-

Comandos desabilitados pelo DCS for o Redis 5.0

A lista a seguir apresenta os comandos suportados pelo DCS for o Redis 5.0.

Tabela 5-21 Comandos do Redis desabilitados em instâncias do Redis 5.0 de nó único e de DCS principal/em espera

Chaves	Servidor
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	Comandos DEBUG
-	COMMAND
-	SAVE
-	BGSAVE
-	BGREWRITEAOF
-	SYNC
-	PSYNC

Tabela 5-22 Comandos do Redis desativados em instâncias do Redis 5.0 do Cluster de Proxy DCS

Chaves	Servidor	Conjunto Ordenado	Aglomerado
MIGRATE	BGREWRITEAOF	-	READONLY
MOVE	BGSAVE	-	READWRITE
RANDOMKEY	Comandos CLIENT	-	-
WAIT	DEBUG OBJECT	-	-
-	DEBUG SEGFAULT	-	-
-	LASTSAVE	-	-
-	PSYNC	-	-
-	SAVE	-	-
-	SHUTDOWN	-	-
-	SLAVEOF	-	-
-	Comandos LATENCY	-	-

Chaves	Servidor	Conjunto Ordenado	Aglomerado
-	Comandos MODULE	-	-
-	LOLWUT	-	-
-	SWAPDB	-	-
-	REPLICAOF	-	-
-	SYNC	-	-

Tabela 5-23 Comandos do Redis desativados em instâncias do Redis 5.0 do Cluster de Proxy DCS

Chaves	Servidor	Aglomerado
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS
-	LASTSAVE	CLUSTER ADDSLOTS
-	Comandos DEBUG	CLUSTER DELSLOTS
-	COMMAND	CLUSTER SETSLOT
-	SAVE	CLUSTER BUMPEPOCH
-	BGSAVE	CLUSTER SAVECONFIG
-	BGREWRITEAOF	CLUSTER FORGET
-	SYNC	CLUSTER REPLICATE
-	PSYNC	CLUSTER COUNT-FAILURE-REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

Tabela 5-24 Comandos do Redis desativados em instâncias de divisão de leitura/gravação do DCS Redis 5.0

Aglomerado	Chaves	Servidor
READONLY	MIGRATE	BGREWRITEAOF
READWRITE	WAIT	BGSAVE

Aglomerado	Chaves	Servidor
-	-	DEBUG OBJECT
-	-	DEBUG SEGFAULT
-	-	LASTSAVE
-	-	LOLWUT
-	-	MODULE LIST/LOAD/ UNLOAD
-	-	PSYNC
-	-	REPLICAOF
-	-	SAVE
-	-	SHUTDOWN [NOSAVE| SAVE]
-	-	SLAVEOF
-	-	SWAPDB
-	-	SYNC

5.4 Comandos do Redis 6.0 (OBT)

O DCS for Redis 6.0 é compatível com protocolos e comandos de código aberto. A edição básica é baseada no Redis 6.2.7 e a edição profissional é baseada no KeyDB 6.0.16.

Esta seção descreve a compatibilidade do DCS for Redis 6.0 com comandos KeyDB, incluindo comandos suportados e desabilitados.

Para obter mais informações sobre a sintaxe de comandos, visite o **site oficial do KeyDB**.

As instâncias do DCS Redis são compatíveis com a maioria dos comandos do Redis. Qualquer cliente compatível com o protocolo Redis pode acessar o DCS.

- Por motivos de segurança, alguns comandos do Redis são desativados no DCS, conforme listado em **Comandos desabilitados pelo DCS for o Redis 6.0**.
- Alguns comandos do Redis são suportados por instâncias de DCS de cluster para operações de várias chaves no mesmo slot. Para mais detalhes, consulte **Restrições de Comando**.
- Alguns comandos do Redis têm restrições de uso, que são descritas em **Outras restrições de uso de comandos**.

Comandos suportados pelo DCS for Redis 6.0

A lista a seguir apresenta os comandos suportados pelo DCS for o Redis 6.0.

Tabela 5-25 Comandos suportados pelo DCS for Redis 6.0 (1)

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENT KILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT LIST
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT GETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENTE SETNAME
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIG GET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	HSTRLEN	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	SWAPDB
OBJECT	PSETEX	-	RPUSH	SPOP	ZSCAN	MEMORY
PEXPIREAT	SET	-	RPUSHX	-	ZRANGEBYLEX	CONFIG
PEXPIRE	SETBIT	-	LPUSH	-	ZLEXCOUNT	-

Chaves	String	Hash	Lista	Conjunto	Conjunto Ordenado	Servidor
-	SETEX	-	-	-	ZPOPMIN	-
-	SETNX	-	-	-	ZPOPMAX	-
-	SETRANGE	-	-	-	ZREMRANGEBYSCORE	-
-	STRLEN	-	-	-	ZREM	-
-	BITFIELD	-	-	-	-	-

Tabela 5-26 Comandos suportados pelo DCS for Redis 6.0 (2)

HyperLoglog	Pub/Sub	Transações	Ligação	Scripting	Geográficos	Córrego
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD	XACK
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH	XADD
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XCLAIM
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST	XDEL
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS	XGROUP
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER	XINFO
-	-	-	-	-	-	XLEN
-	-	-	-	-	-	XPENDING
-	-	-	-	-	-	XRANGE
-	-	-	-	-	-	XREAD
-	-	-	-	-	-	XREADGROUP
-	-	-	-	-	-	XREVRANGE
-	-	-	-	-	-	XTRIM

Comandos desabilitados pelo DCS for o Redis 6.0

A lista a seguir apresenta os comandos suportados pelo DCS for o Redis 6.0.

Tabela 5-27 Comandos do Redis desativados por instâncias principal/em espera do DCS Redis 6.0

Chaves	Servidor
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	Comandos DEBUG
-	COMMAND
-	SAVE
-	BGSAVE
-	BGREWRITEAOF
-	SYNC
-	PSYNC

5.5 Comandos da CLI da Web

A CLI da Web é uma ferramenta de linha de comando fornecida no console do DCS. Esta seção descreve a compatibilidade da CLI da Web com comandos do Redis, incluindo comandos suportados e desabilitados. Atualmente, apenas os DCS for Redis 4.0, 6.0, e 5.0 oferecem suporte à Web CLI.

NOTA

- Atualmente, nenhum parâmetro de comando na Web CLI oferece suporte ao chinês. Espaços não podem ser incluídos em chaves e valores.
- Se o valor estiver vazio, é devolvido **nil** após a execução do comando **GET**.

Comandos suportados na CLI da Web

A seguir, lista os comandos suportados ao usar a CLI da Web. Para obter detalhes sobre a sintaxe do comando, visite o **site oficial do Redis**. Por exemplo, para exibir detalhes sobre o comando **SCAN**, digite **SCAN** na caixa de pesquisa **desta página**.

Tabela 5-28 Comandos suportados pela CLI da Web (1)

Chaves	String	Lista	Conjunto	Conjunto Ordenado	Servidor
DEL	APPEND	RPUSH	SADD	ZADD	FLUSHALL

Chaves	String	Lista	Conjunto	Conjunto Ordenado	Servidor
OBJECT	BITCOUNT	RPUSHX	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	LLEN	SINTERSTORE	ZRANGEBYSCORE	CLIENT KILL
PTTL	GET	LPOP	SISMEMBER	ZRANK	CLIENT LIST
RANDOMKEY	GETRANGE	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT GETNAME
RENAME	GETSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT SETNAME
RENAMENX	INCR	LREM	SPOP	ZREVRANGE	CONFIG GET
SCAN	INCRBY	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	MONITOR
SORT	INCRBYFLOAT	LTRIM	SREM	ZREVRANK	SLOWLOG
TTL	MGET	RPOP	SUNION	ZSCORE	ROLE
TYPE	MSET	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	SWAPDB
-	MSETNX	RPOPLPUSH	SSCAN	ZINTERSTORE	MEMORY
-	PSETEX	-	SPOP	ZSCAN	-
-	SET	-	-	ZRANGEBYLEX	-
-	SETBIT	-	-	ZLEXCOUNT	-
-	SETEX	-	-	-	-
-	SETNX	-	-	-	-
-	SETRANGE	-	-	-	-
-	STRLEN	-	-	-	-

Chaves	String	Lista	Conjunto	Conjunto Ordenado	Servidor
-	BITFIELD	-	-	-	-

Tabela 5-29 Comandos suportados pela CLI da Web (2)

Hash	HyperLogLog	Ligação	Scripting	Geográficos
HDEL	PFADD	AUTH	EVAL	GEOADD
HEXISTS	PFCOUNT	ECHO	EVALSHA	GEOHASH
HGET	PFMERGE	PING	SCRIPT EXISTS	GEOPOS
HGETALL	-	QUIT	SCRIPT FLUSH	GEODIST
HINCRBY	-	-	SCRIPT KILL	GEORADIUS
HINCRBYFLOAT	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER
HKEYS	-	-	-	-
HMGET	-	-	-	-
HMSET	-	-	-	-
HSET	-	-	-	-
HSETNX	-	-	-	-
HVALS	-	-	-	-
HSCAN	-	-	-	-
HSTRLEN	-	-	-	-

Comandos desabilitados na CLI da Web

O seguinte lista os comandos desabilitados ao usar a CLI da Web.

Tabela 5-30 Comandos do Redis desativados na ILC da Web para instâncias de nó único e principal/em espera (1)

Chaves	Servidor	Transações	Pub/Sub
MIGRATE	SLAVEOF	UNWATCH	PSUBSCRIBE
WAIT	SHUTDOWN	REPLICAOF	PUBLISH
DUMP	Comandos DEBUG	DISCARD	PUBSUB

Chaves	Servidor	Transações	Pub/Sub
RESTORE	CONFIG SET	EXEC	PUNSUBSCRIBE
-	CONFIG REWRITE	MULTI	SUBSCRIBE
-	CONFIG RESETSTAT	WATCH	UNSUBSCRIBE
-	SAVE	-	-
-	BGSAVE	-	-
-	BGREWRITEAOF	-	-
-	COMMAND	-	-
-	KEYS	-	-
-	MONITOR	-	-
-	SYNC	-	-
-	PSYNC	-	-
-	ACL	-	-

Tabela 5-31 Comandos do Redis desativados na ILC da Web para instâncias de nó único e principal/em espera (2)

Lista	Ligação	Conjunto Ordenado
BLPOP	SELECT	BZPOPMAX
BRPOP	-	BZPOPMIN
BLMOVE	-	BZMPOP
BRPOPLPUSH	-	-
BLMPOP	-	-

Tabela 5-32 Comandos do Redis desativados na CLI da Web para instâncias de cluster do Redis (1)

Chaves	Servidor	Transações	Aglomerado
MIGRATE	SLAVEOF	UNWATCH	CLUSTER MEET
WAIT	SHUTDOWN	REPLICAOF	CLUSTER FLUSHSLOTS
DUMP	Comandos DEBUG	DISCARD	CLUSTER ADDSLOTS
RESTORE	CONFIG SET	EXEC	CLUSTER DELSLOTS

Chaves	Servidor	Transações	Aglomerado
-	CONFIG REWRITE	MULTI	CLUSTER SETSLOT
-	CONFIG RESETSTAT	WATCH	CLUSTER BUMPEPOCH
-	SAVE	-	CLUSTER SAVECONFIG
-	BGSAVE	-	CLUSTER FORGET
-	BGREWRITEAOF	-	CLUSTER REPLICATE
-	COMMAND	-	CLUSTER COUNT-FAILURE-REPORTS
-	KEYS	-	CLUSTER FAILOVER
-	MONITOR	-	CLUSTER SET-CONFIG-EPOCH
-	SYNC	-	CLUSTER RESET
-	PSYNC	-	-
-	ACL	-	-

Tabela 5-33 Comandos do Redis desativados na CLI da Web para instâncias de cluster do Redis (2)

Pub/Sub	Lista	Ligação	Conjunto Ordenado
PSUBSCRIBE	BLPOP	SELECT	BZPOPMAX
PUBLISH	BRPOP	-	BZPOPMIN
PUBSUB	BLMOVE	-	BZMPOP
PUNSUBSCRIBE	BRPOPLPUSH	-	-
SUBSCRIBE	BLMPOP	-	-
UNSUBSCRIBE	-	-	-

5.6 Comandos do Memcached (indisponível em breve)

NOTA

O DCS for Memcached está prestes a ficar indisponível e não é mais vendido em algumas regiões. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.

O Memcached suporta o protocolo de texto baseado em TCP e o protocolo binário. Qualquer cliente compatível com um protocolo Memcached pode acessar instâncias de DCS.

Protocolo de texto memcached

O protocolo de texto Memcached usa texto ASCII para transferir comandos, ajudando você a compilar clientes e depurar problemas. As instâncias do Memcached DCS podem até mesmo ser conectadas diretamente usando Telnet.

Comparado com o protocolo binário Memcached, o protocolo de texto Memcached é compatível com mais clientes de código aberto, mas o protocolo de texto não suporta autenticação.

NOTA

Os clientes podem usar o protocolo de texto Memcached para acessar instâncias do Memcached DCS somente se o acesso sem senha estiver habilitado. O acesso sem senha significa que o acesso às instâncias do DCS Memcached não será protegido por nome de usuário e senha, e qualquer cliente do Memcached que atenda às regras do grupo de segurança na mesma VPC poderá acessar as instâncias. Permitir o acesso sem senha representa riscos de segurança. Tenha cuidado ao habilitar o acesso sem senha.

Tabela 5-34 lista os comandos suportados pelo protocolo de texto Memcached e descreve se esses comandos são suportados pelas instâncias do Memcached DCS.

Tabela 5-34 Comandos suportados pelo protocolo de texto Memcached

Command	Description	Supported by DCS
Adicionar	Adiciona dados.	Sim
Definir	Define dados, incluindo a adição ou modificação de dados.	Sim
Substituir	Substitui dados.	Sim
Anexar	Adiciona dados após o valor da chave especificada.	Sim
prependência	Adiciona dados antes do valor de uma chave especificada.	Sim
cas	Verifica e define dados.	Sim
Obter	Consulta dados.	Sim
Obtém	Consulta detalhes de dados.	Sim
Excluir	Exclui dados.	Sim
incr	Adiciona a quantia especificada ao contador solicitado.	Sim
decr	Remove a quantia especificada para o contador solicitado.	Sim
toque	Atualiza o tempo de expiração dos dados existentes.	Sim
quito	Fecha a conexão.	Sim

Command	Description	Supported by DCS
rubor_todos	Invalida todos os dados existentes. **NOTA** O valor da opção de atraso (se houver) deve ser **0**.	Sim
versão	Consulta informações sobre a versão do Memcached.	Sim
estatísticas	Gerencia estatísticas de operação. **NOTA** Atualmente, apenas estatísticas básicas podem ser consultadas. Comandos em parâmetros opcionais não podem ser consultados.	Sim
limite_de_cache	Ajusta o limite de memória cache.	Não
lajes	Consulta o uso de estruturas internas de armazenamento.	Não
lru	Gerencia políticas de exclusão de dados expirados.	Não
lru_crawler	Gerencia threads de exclusão de dados expirados.	Não
Detalhamento	Define o nível de verbosidade da saída de registro.	Não
ASSISTA	Inspeciona o que se passa internamente.	Não

Protocolo Binário Memcached

O protocolo binário Memcached codifica comandos e operações em estruturas específicas antes de enviá-los. Os comandos são representados por cadeias de caracteres predefinidas.

O protocolo binário Memcached fornece mais recursos, mas menos clientes do que o protocolo de texto Memcached. O protocolo binário Memcached é mais seguro do que o protocolo de texto Memcached, pois suporta adicionalmente a autenticação simples e a autenticação da camada de segurança (SASL).

Tabela 5-35 A lista os comandos suportados pelo protocolo binário do Memcached e descreve se esses comandos são suportados pelas instâncias do Memcached DCS.

Tabela 5-35 Comandos suportados pelo protocolo binário Memcached

Command Code	Command	Description	Supported by DCS
0x00	Obter	Consulta dados.	Sim

Command Code	Command	Description	Supported by DCS
0x01	Definir	Define dados, incluindo a adição ou modificação de dados.	Sim
0x02	Adicionar	Adiciona dados.	Sim
0x03	SUBSTITUIÇÃO	Substitui dados.	Sim
0x04	Excluir	Exclui dados.	Sim
0x05	INCREMENTO	Adiciona a quantia especificada ao contador solicitado.	Sim
0x06	DECRETO	Remove a quantia especificada para o contador solicitado.	Sim
0x07	QUITO	Fecha a conexão.	Sim
0x08	FLUSH	Invalida todos os dados existentes. **NOTA** O valor da opção de atraso (se houver) deve ser **0**.	Sim
0x09	GETQ (tradução)	Consulta dados. O cliente não receberá qualquer resposta em caso de falha.	Sim
0x0a	NOOP	Instrução sem operação, equivalente a ping.	Sim
0x0b	versão	Consulta informações sobre a versão do Memcached.	Sim
0x0c	GETK (tradução)	Consulta dados e adiciona uma chave no pacote de resposta.	Sim
0x0d	GETKQ	Consulta dados e retorna uma chave. O cliente não receberá qualquer resposta em caso de falha.	Sim
0x0e	Anexar	Adiciona dados após o valor da chave especificada.	Sim
0x0f	PREPENDO	Adiciona dados antes do valor de uma chave especificada.	Sim

Command Code	Command	Description	Supported by DCS
0x10	STAT	Consulta estatísticas de instâncias do Memcached do DCS. **NOTA** Atualmente, apenas estatísticas básicas podem ser consultadas. Comandos em parâmetros opcionais não podem ser consultados.	Sim
0x11	SETQ	Define dados, incluindo a adição ou modificação de dados. O comando **SETQ** somente retorna uma resposta em caso de falhas. O cliente não receberá qualquer resposta em caso de sucesso.	Sim
0x12	ADDQ	Adiciona dados. Ao contrário do comando **ADD**, o comando **ADDQ** só devolve uma resposta em caso de falhas. O cliente não receberá qualquer resposta em caso de sucesso.	Sim
0x13	SUBSTITUIÇÃO	Substitui dados. Ao contrário do comando **REPLACE** , o comando **REPLACEQ** só devolve uma resposta em caso de falhas. O cliente não receberá qualquer resposta em caso de sucesso.	Sim
0x14	DELETEQ	Exclui dados. Ao contrário do comando **DELETE** , o comando **DELETEQ** só retorna uma resposta em caso de falhas. O cliente não receberá qualquer resposta em caso de sucesso.	Sim
0x15	INCREMENTQ	Adiciona a quantia especificada ao contador solicitado. Diferentemente do comando **INCREMENT** , o comando **INCREMENTQ** só devolve uma resposta em caso de falhas. O cliente não receberá qualquer resposta em caso de sucesso.	Sim

Command Code	Command	Description	Supported by DCS
0x16	DECRETOQ	Remove a quantia especificada para o contador solicitado. Diferentemente do comando **DECREMENT**, o comando **DECREMENTQ** só devolve uma resposta em caso de falhas. O cliente não receberá qualquer resposta em caso de sucesso.	Sim
0x17	QUITQ	Fecha a conexão.	Sim
0x18	FLUSHQ	Limpa dados e não retorna nenhuma informação. **NOTA** O valor da opção de atraso (se houver) deve ser **0**.	Sim
0x19	APÊNDICE	Adiciona dados após o valor da chave especificada. Ao contrário do comando **APPEND** , o comando **APPENDQ** só devolve uma resposta em caso de falhas. O cliente não receberá qualquer resposta em caso de sucesso.	Sim
0x1a	PREPENDQ	Adiciona dados antes do valor de uma chave especificada. Ao contrário do comando **PREPEND**, o comando **PREPENDQ** só devolve uma resposta em caso de falhas. O cliente não receberá qualquer resposta em caso de sucesso.	Sim
0x1c	TOQUE	Atualiza o tempo de expiração dos dados existentes.	Sim
0x1d	GAT	Consulta dados e atualiza o tempo de expiração dos dados existentes.	Sim
0x1e	GATQ	Consulta dados e retorna uma chave. O cliente não receberá qualquer resposta em caso de falha.	Sim
0x23	GATK	Consulta dados, adiciona uma chave no pacote de resposta e atualiza o tempo de expiração dos dados existentes.	Sim

Command Code	Command	Description	Supported by DCS
0x24	GATKQ	Consulta dados, retorna uma chave e atualiza o tempo de expiração dos dados existentes. O cliente não receberá qualquer resposta em caso de falha.	Sim
0x20	SASL_LIST_MECHS	Pergunta ao servidor quais mecanismos de autenticação SASL ele suporta.	Sim
0x21	SASL_AUTH	Inicia a autenticação SASL.	Sim
0x22	SASL_STEP	Outras etapas de autenticação são necessárias.	Sim

5.7 Restrições de Comando

Alguns comandos do Redis são suportados por instâncias de DCS de cluster para operações de várias chaves no mesmo slot. Para mais detalhes, consulte **Tabela 5-36**.

Tabela 5-37 e **Tabela 5-38** liste os comandos restritos para Cluster de proxy e instâncias de divisão de leitura/gravação do DCS Redis 4.0.

Tabela 5-39 e **Tabela 5-40** liste os comandos restritos para Cluster de proxy e instâncias de divisão de leitura/gravação do DCS Redis 5.0.

Tabela 5-36 Comandos do Redis restritos em instâncias de DCS do Cluster do Redis

Categoria	Descrição
Set	
SINTER	Retorna os membros do conjunto resultantes da interseção de todos os conjuntos fornecidos.
SINTERSTORE	Igual a **SINTER**, mas em vez de retornar o conjunto de resultados, ele é armazenado no *destino*.
SUNION	Retorna os membros do conjunto resultantes da união de todos os conjuntos fornecidos.
SUNIONSTORE	Igual a **SUNION**, mas em vez de retornar o conjunto de resultados, ele é armazenado no *destino*.
SDIFF	Retorna os membros do conjunto resultantes da diferença entre o primeiro conjunto e todos os conjuntos sucessivos.
SDIFFSTORE	Igual a **SDIFF**, mas em vez de retornar o conjunto de resultados, ele é armazenado no *destino*.

Categoria	Descrição
SMOVE	Move o **member** do conjunto na **source** para o conjunto no *destino*.
Sorted Set	
ZUNIONSTORE	Calcula a união de conjuntos *numéricos* ordenados dados pelas chaves especificadas.
ZINTERSTORE	Calcula a interseção de *numkeys* conjuntos ordenados dados pelas chaves especificadas.
HyperLogLog	
PFCOUNT	Retorna a cardinalidade aproximada calculada pela estrutura de dados HyperLogLog armazenada na variável especificada.
PFMERGE	Mescla vários valores HyperLogLog em um valor exclusivo.
Keys	
RENAME	Renomeia a *chave* para *newkey*.
RENAMENX	Renomeia *key* para *newkey* se *newkey* ainda não existir.
BITOP	Executa uma operação bit a bit entre várias chaves (contendo valores de string) e armazena o resultado na chave de destino.
RPOPLPUSH	Retorna e remove o último elemento (cauda) da lista armazenada na fonte, e empurra o elemento no primeiro elemento (cabeça) da lista armazenada no *destino*.
String	
MSETNX	Define as chaves fornecidas para seus respectivos valores.

NOTA

Ao executar comandos que levam muito tempo para serem executados, como **FLUSHALL**, as instâncias DCS podem não responder a outros comandos e podem mudar para o estado defeituoso. Após o término da execução do comando, a instância retornará ao normal.

Tabela 5-37 Comandos do Redis restritos para instâncias do Cluster de Proxy DCS Redis 4.0

Categoria	Comando	Restrição
Conjuntos	SMOVE	Para uma instância de Cluster de Proxy, as chaves de origem e de destino devem estar no mesmo slot.
Geográficos	GEORADIUS	● Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	GEORADIUSBYMEMBER	

Categoria	Comando	Restrição
	GEOSEARCHSTORE	● Para uma instância de cluster de proxy com vários bancos de dados, a opção **STORE** não é suportada.
Conexão	CLIENTE KILL	● Somente os dois formatos a seguir são suportados: – CLIENT KILL ip:porta – CLIENT KILL ADDR ip:porta ● O campo **id** tem um valor aleatório e não atende ao requisito idc1<idc2→Tc1<Tc2.
	CLIENT LIST	● Somente os dois formatos a seguir são suportados: – CLIENT LIST – CLIENT LIST [TYPE normal|principal|replica|pubsub] ● O campo **id** tem um valor aleatório e não atende ao requisito idc1<idc2→Tc1<Tc2.

Categoria	Comando	Restrição
	SELECT índice	Multi-DB de instâncias de cluster de proxy pode ser implementado alterando as chaves. Esta solução não é recomendada. Restrições no suporte a vários bancos de dados para uma instância de cluster de proxy: 1. O armazenamento de backend reescreve chaves com base em certas regras. As chaves no arquivo RDB exportado não são as chaves originais, mas ainda podem ser acessadas por meio do protocolo Redis. 2. O comando **FLUSHDB** apaga as teclas uma a uma, o que leva muito tempo. 3. **SWAPDB** não é suportado. 4. O comando **INFO KEYSPACE** não devolve dados de vários bancos de dados. 5. O comando **DBSIZE** é demorado. Não o use no código. 6. Se forem utilizados vários bancos de dados, o desempenho dos comandos **KEYS** e **SCAN** se deteriora em até 50%. 7. Scripts LUA não suportam o uso de vários bancos de dados. 8. O comando **RANDOMKEY** não suporta o uso de múltiplos bancos de dados. 9. Por padrão, multi-DB está desabilitado. Antes de ativar ou desativar essa opção para uma instância, limpe os dados da instância.
HyperLogLog	PFCOUNT	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	PFMERGE	
Chaves	RENAME	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	RENAMENX	

Categoria	Comando	Restrição
	SCAN	As instâncias de cluster de proxy não oferecem suporte ao comando **SCAN** em pipelines.
Listas	BLPOP	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	BRPOP	
	BRPOPLPUSH	
Pub/Sub	PSUBSCRIBE	As instâncias de cluster de proxy não oferecem suporte à assinatura de eventos de keyspace, portanto, não haverá falha na assinatura de eventos de keyspace.
Scripting	EVAL	● Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot. ● Quando a função multi-DB está ativada para uma instância de cluster de proxy, o parâmetro **KEYS** é modificado. Preste atenção ao parâmetro **KEYS** usado no script Lua.
	EVALSHA	
Servidor	MEMORY DOCTOR	Para uma instância de cluster de proxy, adicione o *ip*:*port* do nó no final do comando.
	MEMORY HELP	
	MEMORY MALLOC-STATS	
	MEMORY PURGE	
	MEMORY STATS	
	MEMORY USAGE	
	MONITOR	
String	BITOP	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	MSETNX	
de impostos	WATCH	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
Fluxo	XACK	Atualmente, as instâncias do Cluster de Proxy não suportam Streams.
	XADD	
	XCLAIM	
	XDEL	
	XGROUP	

Categoria	Comando	Restrição
	XINFO	
	XLEN	
	XPENDING	
	XRANGE	
	XTRIM	
	XREVRANGE	
	XREAD	
	XREADGROUP GROUP	

Tabela 5-38 Comandos do Redis restritos para divisão de leitura/gravação instâncias do DCS Redis 4.0

Categoria	Comando	Restrição
Conexão	CLIENTE KILL	● Somente os dois formatos a seguir são suportados: – CLIENT KILL ip:porta – CLIENT KILL ADDR ip:porta ● O campo **id** tem um valor aleatório e não atende ao requisito idc1<idc2→Tc1<Tc2.
	CLIENT LIST	● Somente os dois formatos a seguir são suportados: – CLIENT LIST – CLIENT LIST [TYPE normal|principal|replica|pubsub] ● O campo **id** tem um valor aleatório e não atende ao requisito idc1<idc2→Tc1<Tc2.

Tabela 5-39 Comandos do Redis restritos para instâncias do Cluster de Proxy DCS Redis 5.0

Categoria	Comando	Restrição
Conjuntos	SMOVE	Para uma instância de Cluster de Proxy, as chaves de origem e de destino devem estar no mesmo slot.

Categoria	Comando	Restrição
Conjuntos classificados	BZPOPMAX	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	BZPOPMIN	
Geográficos	GEORADIUS	● Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot. ● Para uma instância de Cluster de Proxy no modo multi-DB, a opção **STORE** não é suportada.
	GEORADIUSBYMEMBER	
	GEOSEARCHSTORE	
Conexão	CLIENTE KILL	● Somente os dois formatos a seguir são suportados: – CLIENT KILL ip:porta – CLIENT KILL ADDR ip:porta ● O campo **id** tem um valor aleatório e não atende ao requisito idc1<idc2→Tc1<Tc2.
	CLIENT LIST	● Somente os dois formatos a seguir são suportados: – CLIENT LIST – CLIENT LIST [TYPE normal|principal|replica|pubsub] ● O campo **id** tem um valor aleatório e não atende ao requisito idc1<idc2→Tc1<Tc2.

Categoria	Comando	Restrição
	Índice SELECT	Multi-DB de instâncias de cluster de proxy pode ser implementado alterando as chaves. Esta solução não é recomendada. Restrições no suporte a vários bancos de dados para uma instância de cluster de proxy: 1. O armazenamento de backend reescreve chaves com base em certas regras. As chaves no arquivo RDB exportado não são as chaves originais, mas ainda podem ser acessadas por meio do protocolo Redis. 2. O comando **FLUSHDB** apaga as teclas uma a uma, o que leva muito tempo. 3. **SWAPDB** não é suportado. 4. O comando **INFO KEYSPACE** não devolve dados de vários bancos de dados. 5. O comando **DBSIZE** é demorado. Não o use no código. 6. Se forem utilizados vários bancos de dados, o desempenho dos comandos **KEYS** e **SCAN** se deteriora em até 50%. 7. Scripts LUA não suportam o uso de vários bancos de dados. 8. O comando **RANDOMKEY** não suporta o uso de múltiplos bancos de dados. 9. Por padrão, multi-DB está desabilitado. Antes de ativar ou desativar essa opção para uma instância, limpe os dados da instância.
HyperLogLog	PFCOUNT	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	PFMERGE	
Chaves	RENAME	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	RENAMENX	

Categoria	Comando	Restrição
	SCAN	As instâncias de cluster de proxy não oferecem suporte ao comando **SCAN** em pipelines.
Listas	BLPOP	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	BRPOP	
	BRPOPLPUSH	
Pub/Sub	PSUBSCRIBE	As instâncias de cluster de proxy não oferecem suporte à assinatura de eventos de keyspace, portanto, não haverá falha na assinatura de eventos de keyspace.
Scripting	EVAL	• Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot. • Quando a função multi-DB está ativada para uma instância de cluster de proxy, o parâmetro **KEYS** será modificado. Preste atenção ao parâmetro **KEYS** usado no script Lua.
	EVALSHA	
Servidor	MEMORY DOCTOR	Para uma instância de cluster de proxy, adicione o *ip*:*port* do nó no final do comando.
	MEMORY HELP	
	MEMORY MALLOC-STATS	
	MEMORY PURGE	
	MEMORY STATS	
	MEMORY USAGE	
	MONITOR	
String	BITOP	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
	MSETNX	
de impostos	WATCH	Para uma instância de Cluster de Proxy, todas as chaves transferidas devem estar no mesmo slot.
Fluxo	XACK	Atualmente, as instâncias do Cluster de Proxy não suportam Streams.
	XADD	
	XCLAIM	
	XDEL	
	XGROUP	

Categoria	Comando	Restrição
	XINFO	
	XLEN	
	XPENDING	
	XRANGE	
	XTRIM	
	XREVRANGE	
	XREAD	
	XREADGROUP GROUP	

Tabela 5-40 Comandos do Redis restritos para divisão de leitura/gravação instâncias do DCS Redis 5.0

Categoria	Comando	Restrição
Conexão	CLIENTE KILL	● Somente os dois formatos a seguir são suportados: – CLIENT KILL ip:porta – CLIENT KILL ADDR ip:porta ● O campo **id** tem um valor aleatório e não atende ao requisito idc1<idc2→Tc1<Tc2.
	CLIENT LIST	● Somente os dois formatos a seguir são suportados: – CLIENT LIST – CLIENT LIST [TYPE normal|principal|replica|pubsub] ● O campo **id** tem um valor aleatório e não atende ao requisito idc1<idc2→Tc1<Tc2.
Fluxo	XREAD	A opção **BLOCK** não é suportada.
	XREADGROUP GROUP	

5.8 Outras restrições de uso de comandos

Esta seção descreve restrições em alguns comandos do Redis.

Comando KEYS

No caso de uma grande quantidade de dados armazenados em cache, executar o comando **KEYS** pode bloquear a execução de outros comandos por um longo tempo ou ocupar uma

memória excepcionalmente grande. Portanto, ao executar o comando **KEYS**, descreva o padrão exato e não use chaves **keys ***. Não utilize o comando **KEYS** no ambiente de produção. Caso contrário, o serviço em execução poderá ser afetado.

Comandos no grupo de servidores

- Ao executar comandos que levam muito tempo para serem executados, como **FLUSHALL**, as instâncias DCS podem não responder a outros comandos e podem mudar para o estado defeituoso. Após o término da execução do comando, a instância retornará ao normal.
- Quando o comando **FLUSHDB** ou **FLUSHALL** é executado, a execução de outros comandos de serviço pode ser bloqueada por um longo tempo no caso de uma grande quantidade de dados em cache.

Comandos EVAL e EVALSHA

- Quando o comando **EVAL** ou **EVALSHA** é executado, pelo menos uma chave deve estar contida no parâmetro de comando. Caso contrário, a mensagem de erro "ERR eval/evalsha numkeys deve ser maior que zero no modo de cluster redis" é exibido.
- Quando o comando **EVAL** ou **EVALSHA** é executado, uma instância do DCS Redis de cluster usa a primeira chave para calcular slots. Certifique-se de que as chaves a serem operadas em seu código estejam no mesmo slot. Para mais detalhes, visite o **site oficial do Redis**.
- Para o comando **EVAL**:
  - Aprenda os recursos de script Lua do Redis antes de executar o comando **EVAL**. Para mais detalhes, visite o **site oficial do Redis**.
  - O tempo limite de execução de um script Lua é de 5 segundos. Declarações demoradas, como sono de longa duração e instruções de loop grande, devem ser evitadas.
  - Ao chamar um script Lua, não use funções aleatórias para especificar chaves. Caso contrário, os resultados da execução são inconsistentes nos nós principal e stand-by.

Outras restrições

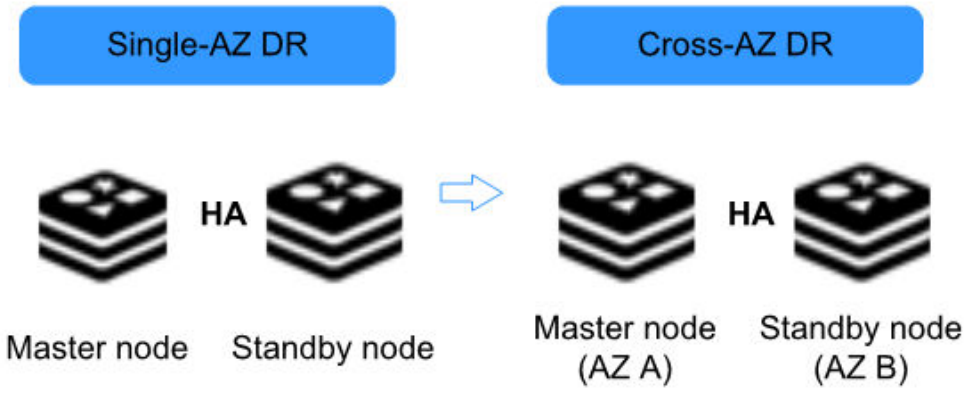
- O limite de tempo para a execução de um comando do Redis é de 15 segundos. Para evitar que outros serviços falhem, um switchover principal/réplica será acionado após o tempo limite de execução do comando.
- As instâncias do Redis do DCS de cluster criadas antes de 10 de julho de 2018 devem ser atualizadas para oferecer suporte aos seguintes comandos:

  SINTER, SDIFF, SUNION, PFCOUNT, PFMERGE, SINTERSTORE, SUNIONSTORE, SDIFFSTORE, SMOVE, ZUNIONSTORE, ZINTERSTORE, EVAL, EVALSHA, BITOP, RENAME, RENAMENX, RPOPLPUSH, MSETNX, SCRIPT LOAD, SCRIPT KILL, SCRIPT EXISTS e SCRIPT FLUSH

6 Recuperação de desastres e solução multi-ativa

Não importa se você usa o DCS como cache de front-end ou como armazenamento de dados de back-end, o DCS está sempre pronto para garantir a confiabilidade dos dados e a disponibilidade do serviço. A figura a seguir mostra a evolução das arquiteturas DCS DR.

Figura 6-1 Evolução da arquitetura DCS DR



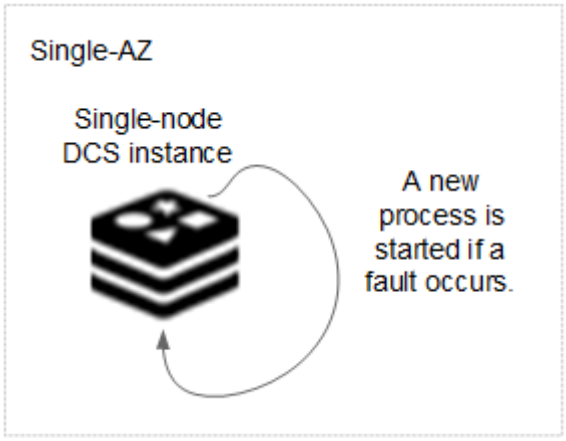
Para atender aos requisitos de confiabilidade de seus dados e serviços, você pode optar por implantar sua instância de DCS em uma única AZ ou entre AZs.

HA Single-AZ dentro de uma região

Implantação Single-AZ significa implantar uma instância dentro de uma sala de equipamentos físicos. O DCS fornece HA de processo/serviço, persistência de dados e políticas de DR em hot em espera para diferentes tipos de instâncias de DCS.

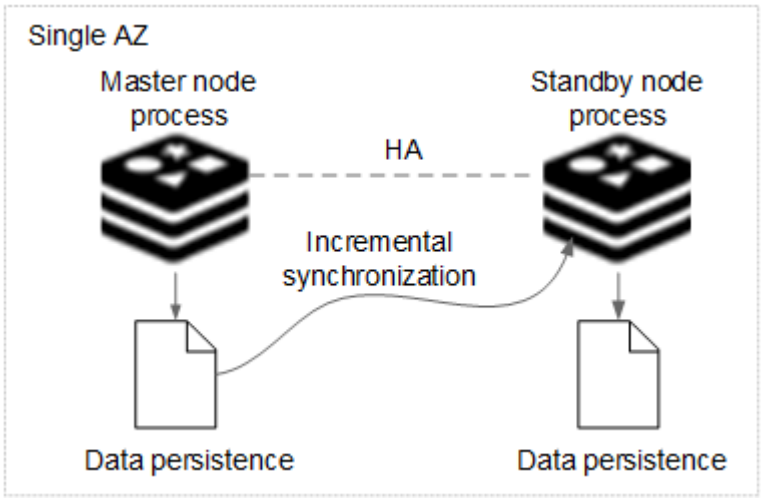
Single-node DCS instance: Quando a DCS detecta uma falha do processo, um processo novo está começado para assegurar o serviço HA.

Figura 6-2 HA para uma instância de DCS de nó único implantada em uma AZ



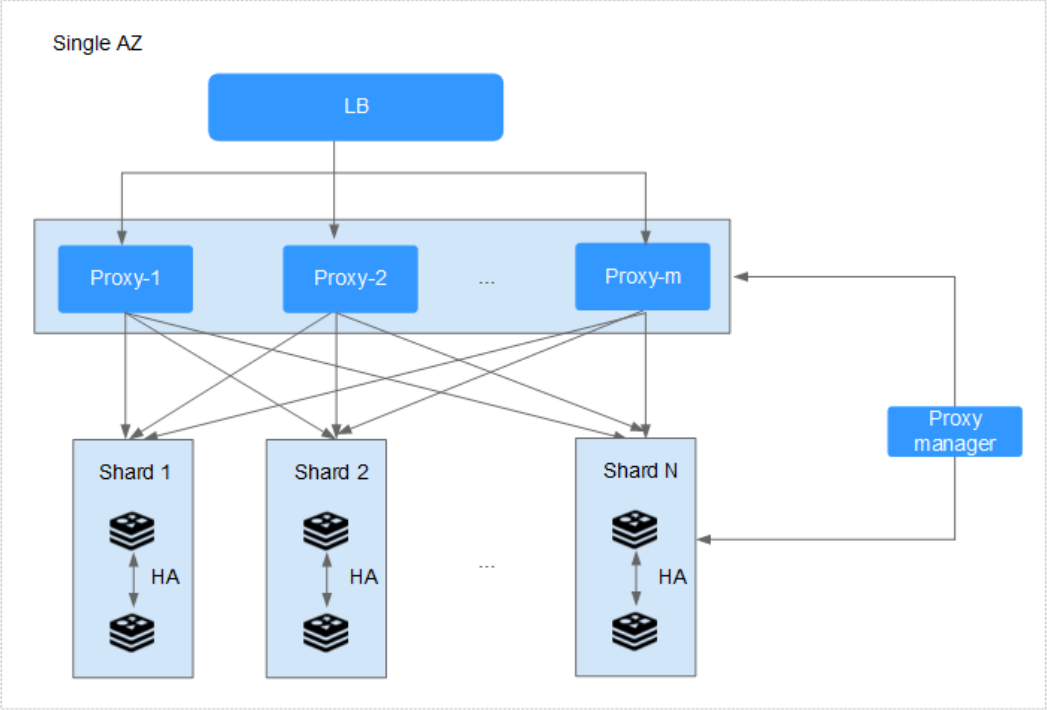
Principal/Em espera DCS instance: Os dados são persistidos no disco no nó principal e sincronizados incrementalmente e persistidos no nó em em espera, obtendo hot em espera e persistência de dados.

Figura 6-3 HA para uma instância de DCS principal/em espera implantada em uma AZ



Cluster DCS instance: Semelhante a uma instância principal/em espera, os dados em cada estilhaço (processo de instância) de uma instância de cluster são sincronizados entre os nós principal e em espera e persistem em ambos os nós.

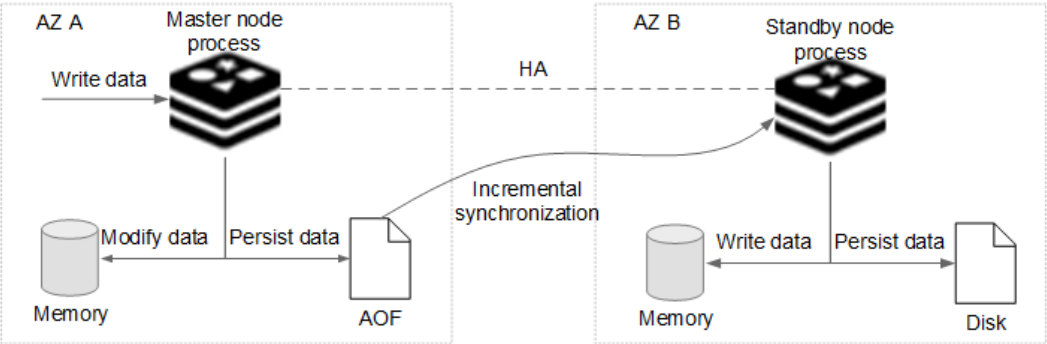
Figura 6-4 HA para uma instância de DCS de cluster implantada em uma AZ



DR Cross-AZ dentro de uma região

Os nós principal e em espera de uma instância de DCS principal/em espera ou cluster podem ser implantados em AZs (em diferentes salas de equipamentos). Fontes de alimentação e redes de diferentes AZs são fisicamente isoladas. Quando ocorre uma falha na AZ em que o nó principal é implantado, o nó em espera se conecta ao cliente e assume as operações de leitura e gravação de dados.

Figura 6-5 Implantação entre AZ de uma instância de DCS principal/em espera



NOTA

Esse mecanismo se aplica de maneira semelhante a uma instância de DCS de cluster, na qual cada estilhaço (processo) é implantado em AZs.

Ao criar uma instância de DCS principal/em espera, selecione uma AZ em espera que seja diferente da AZ principal, conforme mostrado abaixo.

Figura 6-6 Selecionando diferentes AZs

NOTA

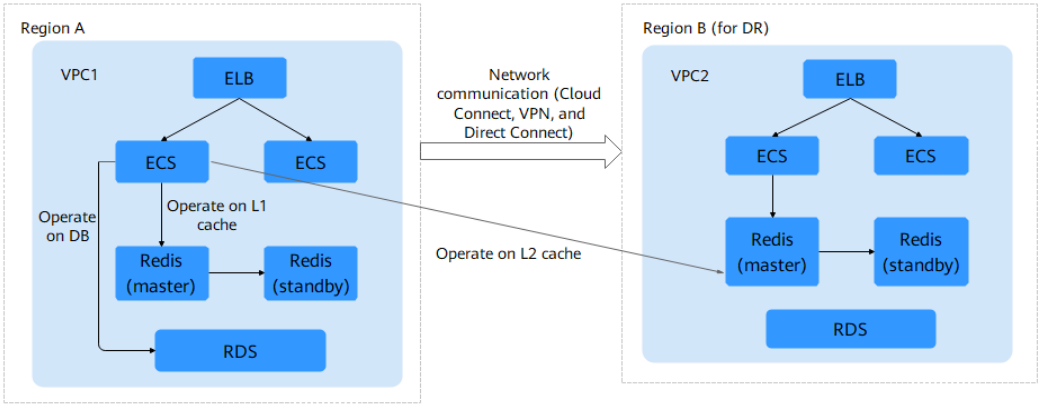
Você também pode implantar seu aplicativo em AZs para garantir a confiabilidade dos dados e a disponibilidade do serviço em caso de interrupção da fonte de alimentação ou da rede.

Multi-Activo Inter-Regiões

Atualmente, o HUAWEI CLOUD DCS não suporta multiativos entre regiões porque o Redis não tem uma solução madura ativo-ativo. **Active-active is different from disaster recovery or principal/em espera HA.**

O Redis ativo-ativo em nuvens ou regiões não pode ser alcançado porque os protocolos de serialização do REdis (RESP) personalizados não são unificados. Se ativo-ativo for necessário, ele pode ser implementado por meio de **dual-write on the application end**.

Figura 6-7 Duplo-escreva no lado da aplicação para conseguir o multi-ativo



Nota:

1. A solução de gravação dupla **cannot ensure cache consistency** (devido a problemas de rede). **Applications** precisam **tolerate cache inconsistency** (definindo o tempo de vida para alcançar a consistência eventual). Se os aplicativos exigem uma consistência de cache forte, essa solução não é adequada. Atualmente, não há solução na indústria para garantir uma forte consistência de cache entre regiões.

2. É aconselhável executar operações no cache L2 entre regiões no modo assíncrono.

7 Diferenças do Cache Engine

7.1 Comparando versões do Redis

Ao criar uma instância do DCS Redis, você pode selecionar a versão do mecanismo de cache e o tipo de instância.

NOTA

O DCS for Redis 3.0 não é mais fornecido. Em vez disso, você pode usar o DCS for Redis 4.0 ou 5.0.

- **Version**

  O DCS suporta ao Redis 5.0, 4.0 e 3.0. **Tabela 7-1** descreve as diferenças entre essas versões. Para obter detalhes sobre os novos recursos do Redis 4.0 e 5.0, consulte **Novos recursos do DCS para Redis 4.0** e **Novos recursos do DCS para Redis 5.0**.

  Tabela 7-1 Diferenças entre as versões do Redis

Características	O Redis 3.0	Redis 4.0 e Redis 5.0	O Redis 6.0
Compatibilidade de código aberto	Redis 3.0.7	Redis 4.0.14 e 5.0.9, respectivamente	Edição básica O Redis 6.2.7 Edição profissional: KeyDB 6.0.16
Modo de implantação da instância	Baseado em VMs	Containerizado com base em servidores físicos	Containerizado com base em servidores físicos
Arquitetura da CPU	x86	x86 e braço	x86

Características	O Redis 3.0	Redis 4.0 e Redis 5.0	O Redis 6.0
Tempo necessário para criar uma instância	3 ‑ 15 minutos Instância do cluster: 10 ‑ 30 minutos	8 segundos	8 segundos
QPS	QPS 100 000 por nó	QPS 100 000 por nó	QPS 300 000 por nó
Acesso à rede pública	Compatível	Incompatível	Incompatível
Conexão de nome de domínio	Suportado em VPC	Suportado em VPC	Suportado em VPC
Gerenciamento de dados visualizados	Incompatível	Fornece Web CLI para acesso ao Redis e gerenciamento de dados.	Fornece Web CLI para acesso ao Redis e gerenciamento de dados.
Tipos de instância	Cluster de nó único, principal/em espera e proxy	Nó único, principal/em espera, cluster de proxy e cluster do Redis	Single-node e principal/em espera
Memória total da instância	Varia de 2 GB a 1024 GB.	As especificações regulares variam de 2 GB a 1024 GB. Especificações pequenas de 128 MB, 256 MB, 512 MB e 1 GB também estão disponíveis para instâncias de nó único e principal/em espera.	4 GB, 8 GB, 16 GB, 32 GB e 64 GB
Expansão/redução de capacidade	Expansão e redução da capacidade online	Expansão e redução da capacidade online	Expansão e redução da capacidade online
Backup e restauração	Suportado para instâncias principal/em espera e de cluster de proxy	Compatível com instâncias de divisão de leitura/gravação, cluster de proxy, cluster do Redis e principal/em espera, cluster de proxy e cluster do Redis	Suportado para instâncias principal/em espera

> **NOTA**
>
> As arquiteturas subjacentes variam de acordo com a versão do Redis. Depois que uma versão do Redis é escolhida, ela não pode ser alterada. Por exemplo, não é possível fazer upgrade de uma instância do DCS Redis 3.0 for o Redis 4.0 ou 5.0. Se você precisar de uma versão superior do Redis, crie uma nova instância que atenda aos seus requisitos e, em seguida, migre os dados da instância antiga para a nova.

- **Instance type**

  O DCS fornece tipos de instância de divisão de nó único, principal/em espera, cluster de proxy, cluster do Redis e leitura/gravação. Para obter detalhes sobre suas arquiteturas e cenários de aplicativos, consulte **Tipos de instância DCS**.

7.2 Comparando Redis e Memcached

Redis e Memcached são bancos de dados in-memory de código aberto populares que são fáceis de usar e fornecem maior desempenho do que bancos de dados relacionais.

Como posso selecionar entre os dois bancos de dados chave-valor?

O Memcached é adequado para armazenar estruturas de dados simples, enquanto o Redis é adequado para armazenar dados mais complexos e maiores que exigem persistência.

Para obter detalhes, consulte a tabela a seguir.

Tabela 7-2 Diferenças entre Redis e Memcached

Item	O Redis	Memcached
Latência	Banco de dados in-memory com latência de sub-milissegundos	Banco de dados in-memory com latência de submilissegundos
Fácil de usar	Sintaxe simples e fácil de usar	Sintaxe simples e fácil de usar
Armazenamento distribuído	Expansão horizontal no modo cluster	Compatível
Cliente multilíngue	Suporta conexões de clientes em mais de 30 idiomas, incluindo Java, C e Python.	Suporta conexões de clientes em mais de 10 idiomas, incluindo Java, C e Python.
Tópico/ Processo	Single-core e single-thread Comunicação single-thread, evitando alternância e contenção de contexto desnecessários E/S sem bloqueio (multiplexação de E/S) é usada para reduzir o consumo de recursos quando vários clientes estão conectados.	Multi-thread e escalável O desempenho do Memcached pode ser melhorado aumentando o número de CPUs. Há uma vantagem de desempenho óbvia no cenário em que o valor da chave é grande.

Item	O Redis	Memcached
Armazenamento persistente	Compatível Cada operação de gravação (adicionar, excluir ou modificar dados) pode ser gravada em disco (arquivo AOF).	Compatível **NOTA** A persistência não é suportada pelo Memcached de código aberto, mas é suportada pelo HUAWEI CLOUD DCS para Memcached.
Estrutura de dados	Suporta estruturas de dados complexas, como hash, lista, conjunto e conjunto classificado, atendendo a vários cenários.	Suporta strings simples.
Suporte a script Lua	Compatível	Incompatível
Backup de snapshot	Compatível Snapshots são gerados periodicamente. Portanto, não há garantia de que os dados não serão perdidos. O Redis bifurca um subprocesso para gerar instantâneos. Quando há uma grande quantidade de dados, o serviço Redis pode ser interrompido por um curto período de tempo.	Incompatível
Migração de dados	Compatível Os dados podem ser copiados e migrados para uma nova instância do Redis por meio da restauração de instantâneos RDB ou da reprodução de arquivos AOF.	Incompatível
Restrição do valor-chave	O valor de uma chave pode ser de até 1 GB.	1 MB
Vários bancos de dados	Uma instância do DCS Redis de nó único ou principal/em espera suporta até 256 bancos de dados do Redis. Uma instância de Cluster de Proxy ou Cluster do Redis suporta apenas uma base de dados, ou seja, DB0.	Incompatível

Com base na comparação anterior, tanto o Redis quanto o Memcached são fáceis de usar e têm alto desempenho. No entanto, o Redis e o Memcached são diferentes em termos de armazenamento de estrutura de dados, persistência, backup, migração e suporte a scripts. É aconselhável selecionar o mecanismo de cache mais apropriado com base em cenários reais de aplicativos.

NOTA

O memcached é adequado para cenários de cache de pequena quantidade de dados estáticos, onde os dados são lidos apenas sem computação e processamento adicionais, por exemplo, trechos de código HTML.

O Redis possui estruturas de dados mais ricas e cenários de aplicativos mais amplos.

8 Infográficos para comparar DCS for Redis com Redis de código aberto

9 Comparando serviços de cache do DCS e open-source

O DCS oferece suporte a instâncias de nó único, principal/em espera e cluster, garantindo alto desempenho de leitura/gravação e acesso rápido aos dados. Ele também suporta várias operações de gerenciamento de instâncias para facilitar sua O&M. Com o DCS, você só precisa se concentrar na lógica do serviço, sem se preocupar com problemas de implantação, monitoramento, escalabilidade, segurança e recuperação de falhas.

O DCS é compatível com o Redis e o Memcached de código aberto e pode ser personalizado com base em suas necessidades. Isso torna os recursos exclusivos do DCS, além das vantagens dos bancos de dados de cache de código aberto.

DCS para Redis vs. Redis de código aberto

Tabela 9-1 Diferenças entre o DCS for Redis e o Redis de código aberto

Características	Redis de código aberto	DCS for Redis
Implantação de serviços	Requer 0,5 a 2 dias para preparar os servidores.	● Uma instância do DCS Redis 3.0 pode ser criada em 5 a 15 minutos. ● As instâncias do DCS Redis 4.0, 6.0, e 5.0 estão em contêiner e podem ser criadas em 8 segundos.
versão	-	Segue de perto as tendências de código aberto e suporta a versão mais recente do Redis. Atualmente, o Redis 6.0, 5.0, 4.0 e 3.0 são suportados.
Segurança	A segurança da rede e do servidor é de responsabilidade do usuário.	● A segurança da rede é garantida usando VPCs e grupos de segurança da HUAWEI CLOUD. ● A confiabilidade dos dados é garantida pela replicação de dados e backup programado.
Desempenho	-	QPS 100.000 por nó. O DCS for Redis 6.0 pode atingir 300.000 de QPS por nó.

Características	Redis de código aberto	DCS for Redis
Monitoramento	Fornece apenas estatísticas básicas.	Fornece mais de 30 métricas de monitoramento e limites e políticas de alarme personalizáveis. ● Várias métricas – As métricas externas incluem o número de comandos, operações simultâneas, conexões, clientes e conexões negadas. – As métricas de uso de recursos incluem uso de CPU, uso de memória física, throughput de entrada de rede e throughput de saída de rede. – As métricas internas incluem o uso da capacidade da instância, bem como o número de chaves, chaves expiradas, canais de PubSub, padrões de keyspace e falhas de keyspace. ● Limites de alarme personalizados e políticas para diferentes métricas para ajudar a identificar falhas de serviço.
Backup e restauração	Compatível	● Suporta backup programado e manual. Arquivos de backup podem ser baixados. ● Os dados de backup podem ser restaurados no console.
Gerenciamento de parâmetros	Sem gerenciamento de parâmetros visualizados	● O gerenciamento de parâmetros visualizados é suportado no console. ● Os parâmetros de configuração podem ser modificados online. ● Os dados podem ser acessados e modificados no console.
Escalabilidade longitudinal	Interrompe serviços e envolve um procedimento complexo, desde modificar a RAM do servidor até modificar a memória do Redis e reiniciar o sistema operacional e os serviços.	● Suporta aumento e redução de escala online sem interromper os serviços. ● As especificações podem ser dimensionadas para cima ou para baixo dentro da faixa disponível com base nos requisitos de serviço.
O&M	Manual O&M	Serviços de O&M de ponta a ponta 24/7

DCS para Memcached vs. Memcached de código aberto

Tabela 9-2 Diferenças entre o DCS para Memcached e o Memcached de código aberto

Características	Memcached de código aberto	DCS for Memcached
Implantação de serviços	Requer 0,5 a 2 dias para preparar os servidores.	Cria uma instância em 5 a 15 minutos.
Segurança	A segurança da rede e do servidor é de responsabilidade do usuário.	● A segurança da rede é garantida usando VPCs e grupos de segurança da HUAWEI CLOUD. ● A confiabilidade dos dados é garantida pela replicação de dados e backup programado.
Desempenho	-	QPS 100.000 por nó
Monitoramento	Fornece apenas estatísticas básicas.	Fornece mais de 30 métricas de monitoramento e limites e políticas de alarme personalizáveis. ● Várias métricas – As métricas externas incluem o número de comandos, operações simultâneas, conexões, clientes e conexões negadas. – As métricas de uso de recursos incluem uso de CPU, uso de memória física, throughput de entrada de rede e throughput de saída de rede. – As métricas internas incluem o uso da capacidade da instância, bem como o número de chaves, chaves expiradas, canais de PubSub, padrões de keyspace e falhas de keyspace. ● Limites de alarme personalizados e políticas para diferentes métricas para ajudar a identificar falhas de serviço.
Backup e restauração	Incompatível	● Suporta backup programado e manual. ● Os dados de backup podem ser restaurados no console.
Manutenção visualizada	Sem gerenciamento de parâmetros visualizados	● O gerenciamento de parâmetros visualizados é suportado no console. ● Os parâmetros de configuração podem ser modificados online.

Características	Memcached de código aberto	DCS for Memcached
Escalabilidade longitudinal	Interrompe serviços e envolve um procedimento complexo, desde modificar a RAM do servidor até modificar a memória do Redis e reiniciar o sistema operacional e os serviços.	● Suporta aumento de escala on-line sem interromper os serviços. ● As especificações podem ser dimensionadas para cima ou para baixo dentro da faixa disponível com base nos requisitos de serviço.
O&M	Manual O&M	Serviços de O&M de ponta a ponta 24/7
Persistência de dados	Incompatível	Suportado para instâncias principal/em espera

10 Observações e restrições

Conta e Quota

- Uma conta pode criar uma instância de DCS somente depois de passar a autenticação de nome real.
- Você só pode criar um certo número de instâncias com certa memória total por padrão. As cotas são diferentes por usuário e por região. Consulte a cota exibida no console. Para obter mais informações sobre cotas, consulte **Quotas**. Para aumentar a cota, envie um **tíquete de serviço**.

Rede

Uma nuvem pública usa VPCs para gerenciar a segurança de rede dos serviços em nuvem.

- O cliente deve ser implantado em um ECS que pertença à mesma VPC que a instância do DCS Redis ou do Memcached.
- Para uma instância do DCS Redis 3.0 ou Memcached, selecione o mesmo grupo de segurança para a instância do DCS e o ECS em que o cliente está implantado. Se a instância do DCS e o ECS pertencerem a grupos de segurança diferentes, adicione regras de entrada e saída para os grupos de segurança. Para obter mais informações sobre como adicionar as regras, consulte **Como configuro um grupo de segurança?**
- Para uma instância do DCS Redis 4.0 ou 5.0, adicione o endereço IP do ECS no qual o cliente está implantado à lista de permissões da instância. Para obter detalhes, consulte **Managing Lista de permissões de endereço IP**.
- Para uma instância do DCS Redis 6.0, adicione o endereço IP do ECS no qual o cliente está implantado à lista de permissões da instância. Para obter detalhes, consulte **Managing Lista de permissões de endereço IP**.

Dimensionamento

Você pode escalar uma instância somente dentro do escopo de cota restante. Se sua cota for insuficiente, envie um **tíquete de serviço** para aumentar a cota.

11 Faturação

O DCS suporta os modos de pagamento por uso e faturamento anual/mensal. Para obter detalhes, consulte **Product Detalhes da definição de preço**.

NOTA

Atualmente, apenas a região CN-Hong Kong suporta faturamento anual/mensal. Se você precisar usar esse modo de cobrança em outras regiões, envie um tíquete de serviço no console para solicitar que a equipe técnica ative a função para você em segundo plano.

Item cobrado

O uso do DCS é cobrado pela especificação da instância do DCS.

Item cobrado	Descrição
instância DCS	Faturamento com base nas especificações da instância DCS.

Nota: O HUAWEI CLOUD DCS cobra com base nas especificações da instância DCS selecionadas, em vez da capacidade real do cache.

Modo de cobrança

O DCS oferece dois modos de cobrança: pay-per-use e anual/mensal. O pagamento por uso é recomendado se você não tiver certeza de suas necessidades futuras de serviço e quiser evitar pagar por recursos não utilizados. No entanto, se você tiver certeza de suas necessidades, anual/mensal será mais barato.

- Anual/mensal: Oferece um desconto maior do que o modo de pagamento por uso e é recomendado para usuários de longo prazo.
- Pay-per-use (por hora): Você pode iniciar e interromper instâncias do DCS conforme necessário e será cobrado com base na duração do uso das instâncias do DCS. O faturamento começa quando uma instância de DCS é criada e termina quando a instância de DCS é excluída. A unidade mínima de tempo é um segundo.
- Você pode alternar entre os modos anual/mensal e pay-per-use.

Configuração alterada

Você pode alterar as especificações de uma instância do DCS Redis ou do Memcached, ou seja, aumentar ou reduzir uma instância e alterar o tipo de instância de nó único para principal/em espera. Depois de alterar com êxito as especificações, a instância é faturada com base nas novas especificações. Para obter detalhes, consulte **Modificando as Especificações de Instância do DCS**.

Renovação

Você pode renovar um pacote de recursos após sua expiração ou pode definir regras de renovação automática para um pacote de recursos. Para obter mais informações sobre a renovação de pacotes de recursos, consulte **Renewal de Renovação**.

Perguntas frequentes

Para obter mais informações sobre faturamento da DCS, consulte as **Perguntas frequentes sobre compras e faturamento**.

12 Gerenciamento de permissões

Se você precisar atribuir permissões diferentes aos funcionários de sua empresa para acessar seus recursos de DCS, o IAM é uma boa opção para o gerenciamento de permissões refinado. O IAM fornece autenticação de identidade, gerenciamento de permissões e controle de acesso, ajudando você a proteger o acesso aos seus recursos da HUAWEI CLOUD.

Com o IAM, você pode usar sua conta da HUAWEI CLOUD para criar usuários do IAM para seus funcionários e atribuir permissões aos usuários para controlar seu acesso a tipos de recursos específicos. Por exemplo, alguns desenvolvedores de software em sua empresa precisam usar recursos de DCS, mas não devem ter permissão para excluir instâncias de DCS ou executar outras operações de alto risco. Nesse cenário, você pode criar usuários do IAM para os desenvolvedores de software e conceder a eles apenas as permissões necessárias para usar os recursos do DCS.

Se sua conta da HUAWEI CLOUD não exigir usuários individuais do IAM para gerenciamento de permissões, pule esta seção.

O IAM pode ser usado gratuitamente. Você paga apenas pelos recursos em sua conta. Para obter mais informações sobre o IAM, consulte **Visão geral do serviço do IAM**.

Permissões de DCS

Por padrão, os novos usuários do IAM não têm permissões atribuídas. Você precisa adicionar um usuário a um ou mais grupos e anexar políticas de permissões ou funções a esses grupos. Os usuários herdam permissões dos grupos aos quais são adicionados e podem executar operações especificadas em serviços de nuvem com base nas permissões.

O DCS é um serviço de nível de projeto implantado e acessado em regiões físicas específicas. Para atribuir permissões de DCS a um grupo de usuários, especifique o escopo como projetos específicos da região e selecione os projetos para que as permissões entrem em vigor. Se **All projects** for selecionado, as permissões entrarão em vigor para o grupo de usuários em todos os projetos específicos da região. Ao acessar o DCS, os usuários precisam alternar para uma região onde foram autorizados a usar esse serviço.

Você pode conceder permissões aos usuários usando funções e políticas.

- Funções: Um tipo de mecanismo de autorização de granulação grosseira que define permissões relacionadas às responsabilidades do usuário. Esse mecanismo fornece apenas um número limitado de funções de nível de serviço para autorização. Ao usar funções para conceder permissões, você também precisa atribuir outras funções das quais as permissões dependem para entrar em vigor. No entanto, as funções não são uma escolha ideal para autorização refinada e controle de acesso seguro.

- Políticas Um tipo de mecanismo de autorização refinado que define as permissões necessárias para realizar operações em recursos de nuvem específicos sob determinadas condições. Esse mecanismo permite uma autorização baseada em políticas mais flexível, atendendo aos requisitos de controle de acesso seguro. Por exemplo, você pode conceder aos usuários do DCS somente as permissões para instâncias de DCS operacionais. A maioria das políticas define permissões com base em APIs. Para as ações de API suportadas pelo DCS, consulte **Permissions suportadas e ações suportadas**.

A Tabela 1 lista todas as funções e políticas definidas pelo sistema suportadas pelos DCS.

Tabela 12-1 Funções e políticas definidas pelo sistema suportadas pelo DCS

Nome da Função/Política	Descrição	Tipo	Dependência
FullAccess da DCS	Todas as permissões para DCS. Os usuários com essas permissões podem operar e usar todas as instâncias do DCS.	Política definida pelo sistema	Nenh
UserAccess da DCS	Permissões comuns de usuário para DCS, excluindo permissões para criar, modificar, excluir instâncias DCS e modificar especificações de instância.	Política definida pelo sistema	Nenh
ReadOnlyAccess da DCS	Permissões somente leitura para DCS. Os usuários com essas permissões só podem exibir os dados da instância do DCS.	Política definida pelo sistema	Nenh
Administrador de DCS	Permissões de administrador para DCS. Os usuários com essas permissões podem operar e usar todas as instâncias do DCS.	Função definida pelo sistema	As funções **Server Administrator** e **Tenant Guest** precisam ser atribuídas no mesmo projeto.

NOTA

A política de **DCS UserAccess** é diferente da política de **DCS FullAccess**. Se você configurar ambos, não poderá criar, modificar, excluir ou escalar instâncias de DCS porque as instruções negar terão precedência sobre as instruções permitidas.

A Tabela 2 lista as operações comuns suportadas por cada política de sistema do DCS. Escolha as políticas de sistema apropriadas de acordo com esta tabela.

Tabela 12-2 Operações comuns suportadas por cada política do sistema

Operação	FullAccess da DCS	UserAccess da DCS	ReadOnlyAccess da DCS	Administrador de DCS
Modificando parâmetros de configuração da instância	√	√	×	√
Exclusão de tarefas em segundo plano	√	√	×	√
Acessando instâncias usando a CLI da Web	√	√	×	√
Modificando o status de execução da instância	√	√	×	√
Expandindo a capacidade da instância	√	×	×	√
Alteração de senhas de instância	√	√	×	√
Modificando instâncias de DCS	√	×	×	√
Executando um switchover principal/em espera	√	√	×	√
Fazendo backup de dados da instância	√	√	×	√
Analisando teclas grandes ou teclas de atalho	√	√	×	√

Operação	FullAccess da DCS	UserAccess da DCS	ReadOnlyAccess da DCS	Administrador de DCS
Criando instâncias do DCS	√	×	×	√
Exclusão de arquivos de backup da instância	√	√	×	√
Atualizando a versão da instância	√	√	×	√
Restaurando dados da instância	√	√	×	√
Redefinindo senhas de instância	√	√	×	√
Migrando dados da instância	√	√	×	√
Baixando dados de backup da instância	√	√	×	√
Exclusão de instâncias de DCS	√	×	×	√
Consultando parâmetros de configuração da instância	√	√	√	√
Consultando logs de restauração da instância	√	√	√	√
Consultando logs de backup da instância	√	√	√	√

Operação	FullAccess da DCS	UserAccess da DCS	ReadOnlyAccess da DCS	Administrador de DCS
Consultando instâncias de DCS	√	√	√	√
Consultando tarefas de plano de fundo da instância	√	√	√	√
Consultando informações de upgrade da instância	√	√	√	√
Consultando todas as instâncias	√	√	√	√
Operando consultas lentas	√	√	√	√

Links úteis

- **Visão geral do serviço IAM**
- **Criando um Usuário e Concedendo Permissões de DCS**
- **Políticas de permissões e ações suportadas**

13 Conceitos básicos

instância DCS

Uma instância é a unidade de recurso mínima fornecida pelo DCS.

Você pode selecionar o mecanismo de cache do Redis ou do Memcached. Os tipos de instância podem ser de nó único, principal/em espera ou cluster. Para cada tipo de instância, várias especificações estão disponíveis.

Para mais detalhes, veja **Especificações da instância de DCS** e **Tipos de instância do DCS**.

Projeto

Os projetos são usados para agrupar e isolar recursos de OpenStack (recursos de computação, recursos de armazenamento e recursos de rede). Um projeto pode ser um departamento ou uma equipe de projeto. Vários projetos podem ser criados para uma conta.

Réplicas

Uma réplica é um nó de uma instância de DCS. Uma instância de réplica única não tem nenhum nó em em espera. Uma instância de duas réplicas tem um nó principal e um nó em espera. Por padrão, cada instância principal/em espera e cada estilhaço de uma instância do Redis Cluster têm duas réplicas. Por exemplo, se o número de réplicas for definido como três para uma instância principal/em espera, a instância terá um nó principal e dois nós em espera.

Acesso à rede pública

Um EIP pode ser vinculado a uma instância do DCS Redis 3.0. Você pode acessar a instância por meio de clientes usando o EIP. O acesso público não é suportado pelas instâncias do DCS Redis 4.0, 6.0, e 5.0.

Stunnel é usado para criptografar o conteúdo de comunicação no acesso à rede pública. O atraso da rede é um pouco maior do que no VPC, portanto, o acesso à rede pública é adequado para comissionamento local na fase de desenvolvimento.

Para obter detalhes, consulte as **Instruções de acesso público**.

Acesso sem senha

As instâncias do DCS Redis e do Memcached podem ser acessadas na VPC sem senhas. A latência é menor porque nenhuma autenticação de senha está envolvida.

Você pode habilitar o acesso sem senha para instâncias que não têm dados confidenciais. Para garantir a segurança dos dados, você não tem permissão para ativar o acesso sem senha para instâncias habilitadas com acesso à rede pública.

Para obter detalhes, consulte **Enabling Acesso sem senha a uma instância do DCS Redis**.

Janela de tempo de manutenção

A janela de tempo de manutenção é o período em que a equipe de serviços do DCS atualiza e mantém a instância.

A manutenção da instância do DCS ocorre apenas uma vez por triprincipal e não interrompe os serviços. Mesmo assim, é aconselhável selecionar um período de tempo em que a demanda de serviço é baixa.

Ao criar uma instância, você deve especificar uma janela de tempo de manutenção, que pode ser modificada após a criação da instância.

Para obter detalhes, consulte **Modifying a Janela de Tempo de Manutenção de uma Instância**

Implantação Cross-AZ

Instâncias principal/em espera são implantadas em diferentes AZs com fontes de alimentação e redes fisicamente isoladas Os aplicativos também podem ser implantados em AZs para obter HA para dados e aplicativos.

Ao criar uma instância principal/em espera de DCS Redis ou Memcached, você pode selecionar **Cross-AZ Deployment** e selecionar uma AZ para o nó stand-by.

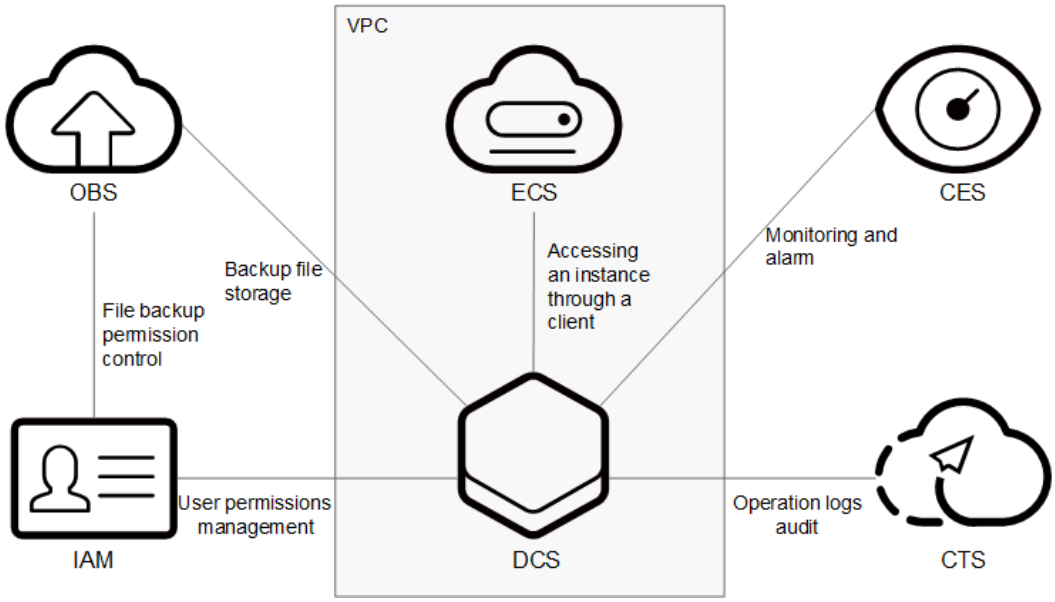
Fragmento

Um estilhaço é uma unidade de gerenciamento de uma instância do DCS Redis de cluster. Cada fragmento corresponde a um processo redis-server. Um cluster consiste em vários estilhaços. Cada estilhaço tem vários slots. Os dados são armazenados de forma distribuída nos slots. O uso de shards aumenta a capacidade do cache e as conexões simultâneas.

14 Serviços relacionados

O DCS é usado em conjunto com outros serviços da HUAWEI CLOUD, incluindo VPC, ECS, IAM, Cloud Eye, CTS e Object Storage Service (OBS).

Figura 14-1 Relações entre DCS e outros serviços



VPC

Uma VPC é um ambiente de rede virtual isolado na HUAWEI CLOUD. Você pode configurar intervalos de endereços IP, sub-redes e grupos de segurança, atribuir EIPs e alocar largura de banda em uma VPC.

O DCS é executado em VPCs. O serviço VPC gerencia EIPs e largura de banda e fornece grupos de segurança. Você pode configurar regras de acesso para grupos de segurança para proteger o acesso ao DCS.

ECS

O Elastic Cloud Server (ECS) é um servidor em nuvem que fornece recursos de computação escaláveis e sob demanda para aplicativos seguros, flexíveis e eficientes.

Você pode acessar e gerenciar suas instâncias de DCS usando um ECS.

IAM

O IAM fornece autenticação de identidade, gerenciamento de permissões e controle de acesso.

Com o IAM, você pode controlar o acesso ao DCS.

Cloud Eye

O Cloud Eye é um serviço de monitoramento seguro, escalável e integrado. Com o Cloud Eye, você pode monitorar seu serviço de DCS e configurar regras e notificações de alarme.

Serviço de Rastreamento de Nuvem (CTS)

O CTS fornece um histórico de operações realizadas em recursos de serviço de nuvem. Com o CTS, você pode consultar, auditar e rastrear operações. Os rastreamentos incluem as solicitações de operação enviadas usando o console de gerenciamento ou APIs abertas e os resultados dessas solicitações.

OBS

O OBS fornece serviço de armazenamento seguro e econômico usando objetos como unidades de armazenamento. Com o OBS, você pode armazenar e gerenciar o ciclo de vida de grandes quantidades de dados.

Você pode armazenar arquivos de backup de instância DCS no OBS.

